Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 25, Sayı 1, 16-27, 2021

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 25, Issue 1, 16-27, 2021

DOI: 10.19113/sdufenbed.661867

Web Uygulama Sızma Testlerinde Kapsam Genişletme İşlemi İçin Metodoloji Geliştirilmesi ve Uygulanması

Mehmet Ali YALÇINKAYA*[1], Ecir Uğur KÜÇÜKSİLLE[2]

[1]Kırşehir Ahi Evran Üniversitesi, Mühendislik Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü, 40100, Kırşehir, Türkiye
[2]Süleyman Demirel Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 32260, Isparta, Türkiye

(Alınış / Received: 19.12.2019, Kabul / Accepted: 08.10.2020, Online Yayınlanma / Published Online: 15.04.2021)

Anahtar Kelimeler
Web uygulama güvenliği,
Sızma testleri,
Siber güvenlik,
Bilgi güvenliği

Özet: Günümüzde, farklı alanlarda hizmet veren tüm kurum ve kuruluşlar, hizmet verdikleri kitlelere daha hızlı ulaşabilmek ve daha etkili hizmet verebilmek adına web uygulamalarını etkin bir şekilde kullanmaktadır. Web uygulamalarının yaygın olarak kullanılması, bu uygulamalara yönelik saldırıların sayısında ve çeşidinde ciddi oranda artışa neden olmuştur. Gerçekleştirilen saldırılar sonrasında oluşan zararın ciddi boyutlara ulaşması nedeniyle, kurum ve kuruluşlar web uygulamalarını belirli periyotlarla sızma testlerine tabi tutmaktadırlar. Sızma testlerinde uzmanlar, web uygulamaları üzerinde çeşitli zafiyetlerin varlığını kontrol etmektedirler. Web uygulamaları üzerinde gerçekleştirilen sızma testlerinde uzmanlara çoğunlukla tek bir URL adresi verilmekte, bu URL adresinden yola çıkarak testlerini gerçekleştirmesi istenmektedir. Bu çalışmada; kullanıcı tercihlerine göre, farklı kaynaklar ve yöntemler kullanarak, test esnasında taranacak URL adreslerinin listesini oluşturan bir kapsam belirleme aracı geliştirilmiştir. Geliştirilen araç, web uygulama sızma testlerinde aktif olarak test kapsamı belirlemede kullanılan 6 farklı araç ile karşılaştırılmış, Httrack aracı ile birlikte en fazla sayıda URL adresini topladığı görülmüştür. Bunun yanında, sunduğu farklı modlarda kapsam tarama modülleri sayesinde, literatürde kullanılan zafiyet tarayıcıların birçoğundan daha modüler ve kapsamlı tarama imkânı sunmaktadır.

Development and Implementation Methodology for Extending Scope in Web Application Penetration Testing

Keywords
Web application security,
Penetration tests,
Cyber security,
Information security

Abstract: Nowadays, all institutions and organizations serving in different fields use web applications effectively in order to reach the masses for serving faster and providing more effective services. The widespread use of web applications has led to a significant increase in the number and type of attacks on these applications. Due to the serious damage caused by the attacks, institutions and organizations subject their web applications to penetration tests periodically. In penetration tests, experts check the presence of various vulnerabilities on web applications. In penetration tests performed on web applications, experts are often given a single URL address and are asked to perform their tests based on this URL. In this study; A scoping tool has been developed that creates a list of URL addresses to be scanned during testing, using different sources and methods, according to user preferences. The developed tool was compared with 6 different tools used to determine the scope of the test actively in web application infiltration tests and it was seen that it collected the maximum number of URL addresses together with the Httrack tool. The developed tool presents a more modular and comprehensive scan facility than many of the vulnerability scanners used in the literature thanks to the scope scanning modules in different modes.

1. Giriş

İnternetin ortaya çıkışından beri eposta, bankacılık, alışveriş gibi kullanıcılar tarafından yoğunlukla tercih edilen ve kullanımı giderek artan web uygulamaları, saldırganların hedefi olmuştur [1]. Günümüzde internet kullanımının ve internet erişimine sahip cihazların oldukça yaygınlaşması, günlük işlerin

*İlgili yazar: mehmetyalcinkaya@ahievran.edu.tr

birçoğunun web uygulamaları üzerinden gerçekleştirilebilmesi, bu uygulamaları insan hayatının en önemli bileşenlerinden biri haline getirmiştir. Ayrıca günümüzde çeşitli alanlardaki kurum ve kuruluşlar için web uygulamaları, herkes tarafından kolaylıkla görüntülenebilen bir vitrin haline gelmiştir. Bunun yanında ülkemiz de dahil olmak üzere, birçok devlet, tüm vatandaşlık hizmetlerini web uygulamaları üzerinden gerçekleştirilebilir hale gelmiştir. Web uygulamalarının bu seviyede kullanıcı sayısına ulaşılması ve çok önemli veriler üzerinde işlem yapması, söz konusu uygulamaları siber saldırganların hedefi haline getirmiştir [2].

Web uygulamaları üzerinde yer alan zafiyetlere yönelik saldırıların artmasıyla, web uygulama sızma testlerine olan ihtiyaç da aynı oranda artmıştır. Web uygulamaları üzerinde gerçekleştirilen sızma testleri; gerçek bir siber saldırı öncesinde, mevcut tüm güvenlik açıklarının tespit edilerek kapatılması için gerçekleştirilmektedir[3]. Bir web uygulaması üzerinde sızma testi gerçekleştirileceği zaman, gerçekleştirilecek en önemli işlemlerden biri sızma testinin kapsamının belirlenmesidir [4]. Gerçekleştirilecek testlerde, kullanıcı isteğine göre sadece bir domain, bir domaine ait alt domain adresleri ya da web uygulaması üzerinde ziyaret edilebilir tüm URL adreslerinin test edilmesi istenebilmektedir. Bu nedenle kullanıcı isteğine göre testin yapılacağı kapsamın tam ve eksiksiz olarak belirlenmesi ve kapsam dahilindeki URL adreslerinin toplanması büyük önem taşımaktadır[2].

Günümüzde web uygulamaları üzerinde gerçekleştirilen testler iki yönteme ayrılmaktadır. İlk yöntemde gerçekleştirilecek tüm testler, herhangi bir araç kullanılmadan manuel olarak gerçekleştirilmektedir. Bu yöntemin dezavantajları; testlerin uzun zaman alması, taranması gereken bazı adreslerin gözden kaçması, testi yapan uzmanın olası düşük bilgi ve tecrübe seviyesinden dolayı başarısız sonuçlar üretilmesidir. Bu dezavantajlar nedeni ile ikinci yöntem, yani; web uygulama sızma testlerinde dinamik analiz tabanlı otomatik zafiyet tarayıcıların kullanımı oldukça yaygınlaşmıştır [2]. Zafiyet tarayıcılar web uygulama sızma testlerinin otomatik olarak gerçekleştirilmesini sağlamaktadır. Otomatikleştirilmiş web uygulaması sızma testinin birçok faydası bulunmaktadır. Zafiyet tarayıcılar sızma testi için gereken zamanı, maliyeti ve kaynağı azaltmakla kalmaz, aynı zamanda test uzamının bilgisine olan ihtiyacı da azaltmaktadır [5].

Bir web uygulama zafiyet tarayıcısının en önemli bileşenlerinden biri ise, crawler olarak adlandırılan kapsam belirleme modülüdür [6]. İyi bir kapsam belirleme aracı, kullanıcılarına farklı seviyelerde tarama imkânı sunmalı, gerçekleştirdiği taramalarda da eksiksiz sonuçlar üretebilmelidir. Bir zafiyet tarayıcının güvenlik açıklarını tespit etmedeki yeteneği, doğrudan kapsam belirleme performansına bağlıdır. Test edilen web uygulamasındaki bir sayfa, form ya da girdi alanı tespit edilemez ise, test edilemez ve olası zafiyetler ortaya çıkarılamaz [7].

Bu çalışmada; bir doktora tezi [2] kapsamında geliştirilmiş, dinamik analiz ve yapay zekâ tabanlı web zafiyet tarayıcısı 'Sansar' 'ın kapsam belirleme modülü işlenmiş, literatürde yer alan diğer zafiyet tarayıcıların sahip olduğu modüller ve sadece bu amaca özel geliştirilmiş araçlar ile karşılaştırılmış, üstün yanları ifade edilmiştir.

Literatürde işlenen konu ile ilgili gerçekleştirilmiş çalışmalar incelendiğinde, mevcut kapsam belirleme modüllerinin incelenmesi ve performanslarının karşılaştırılması üzerine yoğunlaşıldığı görülmüştür. Mantra ve Alaydrus [8] gerçekleştirdikleri çalışmalarda Endonezya da yer alan üniversitelere ait web uygulamaları üzerinde sızma testi gerçekleştirmişler, test esnasında kapsam belirleme için sayfa kaynak kodu üzerindeki URL adreslerini toplayan bir araç kullanmışlardır. Esposito vd [9] çalışmalarında, kapsam belirleme modüllerinin zafiyet tarayıcıların performansını belirlemedeki etkisine değinmişlerdir. Gerçekleştirilen çalışmada kapsam belirleme işlemi için kullanılan ilk yöntem ziyaret edilen sayfanın HTML kodundan URL adreslerinin çıkarılmasıdır. Diğer yöntem ise test edilen web uygulamasının kaynak kodu içerisinde URL adreslerinin aranmasıdır. Akrout vd. [10] çalışmalarında web uygulama sızma testi için bir yaklaşım sunmuşlardır. Söz konusu çalışmada kapsam belirleme işlemi; ziyaret edilen web sayfasının HTML kaynak kodları içerisinden URL adreslerinin tespit edilmesi ve bu işlemin her sayfa için tekrar edilmesi ile gerçekleştirilmiştir. Hillary [11] çalışmasında SQL enjeksiyonu zafiyetinin tespiti için bir tarayıcı geliştirmiştir. Söz konusu çalışmada kapsam belirleme işlemi test edilen URL adresinin parçalanması ve varsa alt domain adreslerinin kontrol edilmesi ile gerçekleştirilmiştir. Mukhopadhyay vd. [12] Nessus ve Metasploit Framework araçlarını kullanarak bir sızma testi metodolojisi geliştirmişlerdir. Söz konusu çalışmada kapsam belirleme işlemi için Wapiti zafiyet tarayıcısnın kapsam belirleme modülü de kullanılmıştır. Dessiatnikoff vd. [13] web uygulamaları üzerindeki SQL enjeksiyonu zafiyetinin tespiti için kendi kapsam belirleme modülüne sahip bir zafiyet tarayıcı geliştirmişlerdir. Idrissi vd. [6] çalışmalarında 11 farklı web uygulama zafiyet tarayıcının zafiyet tespiti performansını karşılaştırmışlardır. Yazarlar ilgili çalışmalarında kapsam belirleme modülünün zafiyet tarayıcı performansına doğrudan etkide bulunduğuna değinmişlerdir. Munoz ve Villalba [14], web uygulama zafiyet yarayıcılar ile ilgili gerçekleştirilmiş çeşitli çalışmaları incelemiş ve bazı önerilerde bulunmuşlardır. Çalışmada incelenen zafiyet tarayıcıların kapsam belirleme modülleri, sayfadan

link toplayan, kullanıcı hareketlerini izleyerek daha sonra uygulayan, girdi alanlarına yazılacak değerleri daha önceden talep eden şeklinde gruplandırılmıştır.

Literatürde web uygulamalarında kapsam genişletme amacı ile yaygın olarak kullanılan çeşitli araçlar da bulunmaktadır. Bunlardan ilki Rapid7 firması tarafından geliştirilen Metasploit Framework' tür [15]. Söz konusu araç kapsam belirleme amacı ile geliştirilmiş 'msfcrawler' isminde bir modüle sahiptir. Kapsam genişletme işlemlerinde kullanılan bir diğer araç HTTrack'tir. Söz konusu araç, URL adresi toplamak için, web uygulaması içerisinde yer alan HTML sayfaları, resimler ve diğer tüm dosyaları yerel dizine indirerek kopyalamaktadır [16]. Web uygulama sızma testlerinde kapsam genişletme işlemlerinde kullanılan bir diğer yaygın araç BlackWidow' dur [17]. Python tabanlı araç, hedef web sitesine ait alt domain, eposta adresleri ve telefon numaraları gibi bilgileri toplamak için kullanılmaktadır. Bunun yanında web uygulama sızma testlerinde yaygın olarak kullanılmakta olan Skipfish [18], Wapiti [19] ve WASCAN [20] araçları da kendi bünyelerinde kapsam genişletme modülleri barındırmaktadırlar.

Literatürde kapsam belirleme amacı ile geliştirilen araçlar; klasik yöntem olan, başlangıç URL adresinin kaynak kodlarından elde edilen diğer URL adreslerini ziyaret etme, ziyaret edilen her bir adresin kaynak kodlarında yeni adresler arama yöntemini kullanmaktadırlar. Nispeten daha gelişmiş olan Wapiti gibi zafiyet tarayıcılar ise ziyaret edilen sayfa üzerindeki formları da doldurarak bir sonraki aşamaya geçmeye çalışmaktadırlar. Bu çalışmada sunulan yöntem ve geliştirilen araç, kullanıcılara 3 farklı türde kapsam belirleme imkanı sağlamaktadır. Kullanıcılar tercihlerine göre sadece alt domain adresi belirleme, kaba kuvvet yöntemi ile gizli URL adreslerini tespit etme, sayfa kaynak kodları ve formlar üzerinden web uygulamasını indeksleme işlemlerini gerçekleştirebilmektedir. 3 metodun aynı anda kullanılmasına imkan veren araç sayesinde, literatüre sızma testlerinin kapsam belirleme adımında kullanılabilecek yeni bir metodoloji kazandırılmıştır.

Bu çalışmanın ikinci bölümünde, çalışmanın temel motivasyonunu oluşturan Web Uygulama Sızma Testleri konusuna değinilmiş, 3. Bölümde web uygulama zafiyet tarayıcılar incelenmiştir. 4. bölümde, geliştirilen kapsam belirleme aracı yapısal olarak işlenmiş, sahip olduğu fonksiyonlar ve kaynak kodlarına yer verilmiştir. Çalışmanın 5. Bölümünde geliştirilen araç kullanılarak gerçekleştirilen kapsam belirleme işlemleri örneklenmiş ve literatürdeki diğer araçlar ile karşılaştırması yapılmıştır. Çalışma, gerçekleştirilen işlemlerin ve elde edilen sonuçların yorumlanması ile tamamlanmıştır.

2. Web Uygulama Sızma Testleri

Sızma testleri; alanında tecrübeli siber güvenlik uzmanları tarafından, kötü niyetli siber saldırganların hedef aldıkları sistemlere verebileceği muhtemel zararları önceden tespit etmek, raporlamak ve gerekli önlemleri almak amacı ile gerçekleştirilen siber saldırılardır [21]. Sızma testlerinde amaç; test edilen sistemler üzerindeki tüm güvenlik zafiyetlerini tespit ederek, ileride gerçekleşmesi muhtemel gerçek bir saldırı öncesinde aksiyon almak ve söz konusu zafiyetleri kapatmaktır [22].

Web uygulamaları; kurumsal kimliğin en önemli göstergelerinden biridir. Bu nedenle; kurumsal bilgi güvenliğini sağlamak amacıyla gerçekleştirilecek sızma testlerinde, test edilmesi gereken ilk kısım; kurum web uygulamasıdır. Web uygulamaları üzerinde gerçekleştirilen sızma testleri; test kapsamına ve test öncesinde uzmana sağlanan bilgilere bağlı olarak üçe ayrılmaktadır. Bunlar; Siyah kutu sızma testleri, beyaz kutu sızma testleri ve gri kutu sızma testleridir [23].

Web uygulamalarına yönelik gerçekleştirilen siyah kutu sızma testlerinde, test uzmanına başlangıç verisi olarak sadece bir URL adresi verilmekte, söz konusu adresten yola çıkarak test işlemlerini gerçekleştirmesi istenmektedir. Siyah kutu test türünde test uzmanı yapacağı ilk işlem, kendisine verilen URL adresinden yola çıkarak, anlaşma kapsamında yer alan ve test edilmesi gereken tüm URL adreslerini toplamak, daha sonra test adımlarını gerçekleştirmektedir. Siyah kutu sızma testlerinde web uygulamaları, yayın esnasında iken dinamik olarak test edilmektedir. Siyah kutu sızma testlerine örnek olarak; sdu.edu.tr adresi ve alt domain adreslerinin test edilmesi istendiği varsayılsın. Bu durumda güvenlik uzmanının kendisine sağlanan tek veriden yola çıkarak tüm alt domain adreslerini bulması gerekmektedir. Test uzmanı alt domain adreslerinin tamamını tespit ettikten sonra zafiyetlere yönelik saldırı denemelerini gerçekleştirecektir.

Web uygulamaları üzerinde gerçekleştirilen beyaz kutu sızma testlerinde ise; kurum tarafından test uzmanına, test edilecek web uygulaması kaynak kodu ile beraber verilmektedir. Bu sayede test uzmanı kaynak kodlar üzerinde yapacağı statik analiz işlemleri ile zafiyete neden olan mantıksal hataları tespit edebilmektedir.

Web uygulama sızma testi türlerinden sonuncusu; gri kutu sızma testleridir. Gri kutu sızma testlerinde; gerçekleştirilecek test senaryoları web uygulamasının iç yapısı baz alınarak belirlendikten sonra, dinamik analiz yöntemi ile gerçekleştirilmektedir. Bu sayede web uygulaması dinamik olarak çalışma anında test edilmesinin yanı

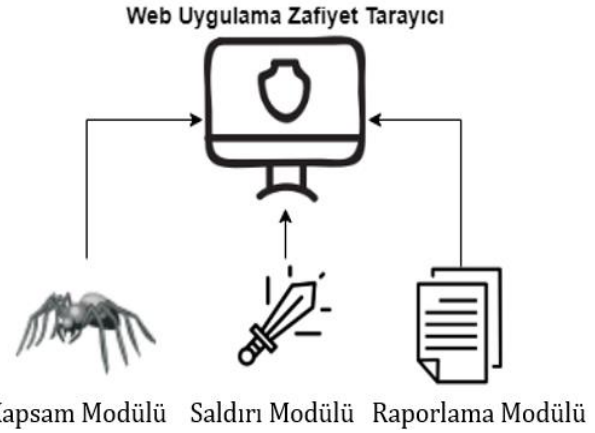
sıra, statik analiz yöntemi sayesinde sahip olduğu iç yapısı da sınanmış olmaktadır [24].

Sızma testi türleri incelendiğinde; özellikle siyah kutu sızma testlerinde, kapsam belirleme amacı ile kullanılacak araçlar büyük önem arz etmektedir. Çünkü söz konusu araçlar sayesinde güvenlik uzmanları bir URL adresinden yola çıkarak tüm web sayfasını gezebilmekte, test edilmesi gereken tüm girdi alanlarını belirleyebilmektedirler. Test kapsamının eksik belirlenmesi demek; test edilmesi gereken URL adresi ve girdi alanlarının pas geçilmesi anlamına gelmektedir. Böyle bir durumda, eğer test edilmeyen URL adreslerinde ve girdi alanlarında bir zafiyet varsa, gerçekleştirilen web uygulama sızma testinin hiçbir anlamı kalmayacaktır. Bu durum tam kapsamlı kurumsal bilgi güvenliğini sağlamanın önünde büyük bir engel teşkil etmektedir. Söz konusu olumsuz durumlar ile karşılaşmamak için, kullanılan kapsam belirleme aracı farklı yöntem ve kaynaklar kullanarak bir web uygulamasını tamamen gezebilmeli, uygulama üzerinde yer alan tüm adreslere erişebilmelidir.

3. Web Uygulama Zafiyet Tarayıcılar

Günümüzde web uygulamalarında güvenliği sağlamanın önemine önceki başlıklarda detaylı olarak değinilmiştir. Web uygulama güvenliği kavramının bu denli önem kazanması; web uygulamaları üzerinde dinamik analiz metodu ile sızma testlerini gerçekleştirecek uzmanlara olan ihtiyacı da arttırmıştır. Güvenlik uzmanları, test edilecek web adreslerinin belirledikten sonra, ilgili sayfalara saldırılar gerçekleştirerek, web uygulaması üzerindeki güvenlik açıklarını tespit etmektedirler. Fakat büyük ölçekli bir kuruma ait bir web uygulamasının birçok alt domain ve URL adreslerine sahip olduğu, söz konusu web sayfaları içerisinde değişken miktarda kullanıcı girdi alanı bulunduğu bir gerçektir. Bu durum göz önüne alındığında, manuel olarak yürütülen test işleminin çok fazla zaman alacağı, istenilen süre içerisinde tamamlanamayacağı bir gerçektir. Ayrıca manuel olarak gerçekleştirilen testlerde; bazı form ve inputların gözden kaçması, bu nedenle testlerinin unutulması, test edilecek zafiyet listesinin kısa olması gibi dezavantajlar ortaya çıkmaktadır. Bu gibi nedenlerden dolayı, dinamik analiz tabanlı web uygulama zafiyet tarayıcılar, web uygulamalarında yer alan güvenlik açıklarını tespit etmek için popüler bir seçim haline gelmiştir [25].

Web uygulama zafiyet tarayıcıları, parametre olarak aldıkları URL adresi üzerinde, çeşitli zafiyetler için geliştirilmiş test metodlarını kullanarak güvenlik testleri gerçekleştirmektedir. Güvenli uzmanları tarafından manuel olarak testleri hızlı bir şekilde ve otomatik olarak gerçekleştiren bu tarayıcılar, test sonrasında tespit ettiği zafiyetleri de kullanıcılarına raporlamaktadırlar. Sahip oldukları kullanım kolaylığı sayesinde, web sızma testinde uzman olmayan kişiler dahi, web sayfalarını söz konusu araçlar ile tarayabilmekte, elde ettiği raporlara göre gerekli önlemleri alabilmektedirler. Web uygulama zafiyet tarayıcıları temel olarak, kapsam belirleme modülü, saldırı (test) modülü ve raporlama modülü olmak üzere 3 ana modülden oluşmaktadır [26]. Şekil 1' de standart bir web uygulama zafiyet tarayıcı ve sahip olduğu modüller gösterilmektedir.



Şekil 1. Web uygulama zafiyet tarayıcı modülleri

Web uygulama zafiyet tarayıcıların sahip olduğu modüllerden ilki olan kapsam belirleme modülü; daha önceki başlıklarda da bahsedildiği üzere, çeşitli kaynak ve yöntemler kullanarak web uygulamasını gezmektedir. Söz konusu modül, gezmiş olduğu her bir URL adresini kaydederek, test edilecek adreslerin listesini oluşturmaktadırlar. Tam kapsamlı bir kapsam belirleme modülü ilk olarak ziyaret ettiği web sayfasının kaynak kodlarını içerisinde farklı HTML taglarına atanmış URL adreslerini toplamalıdır. Söz konusu URL adresleri web uygulamalarında sayfa yönlendirmesi amacıyla paylaşılan linklerdir. Her bir sayfadan toplanan adreslerin daha sonra tek tek ziyaret edilmesi ile web uygulaması büyük ölçüde indekslenecektir. Bu yöntem, mevcut birçok kapsam belirleme aracının web uygulamalarını gezmede kullandığı klasik bir yöntemdir. Söz konusu yöntem web uygulaması içinde gezilmesine olanak sağlasa dahi bir web uygulamasının tam kapsamlı haritasının çıkarılması ve tüm URL adreslerinin elde edilmesi için yeterli değildir. Kullanıcıdan alınan girdilere göre sayfa yönlendirmesi yapan dinamik web uygulamalarının tam kapsamlı gezilebilmesi için, ziyaret edilen web sayfası üzerindeki girdi alanlarının da araç tarafından doldurularak gönderilmesi gerekmektedir. Bu sayede sayfa içerisinde link verilmemiş, sadece doldurulan form aracılığı ile erişilebilen URL adreslerine de ulaşılabilecektir.

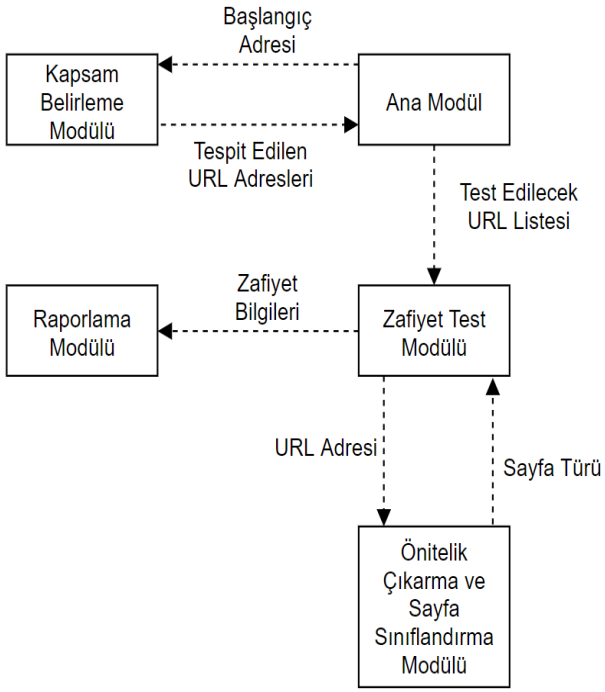
Web uygulama zafiyet tarayıcıların sahip olduğu diğer modüller ise; saldırı ve raporlama modülleridir. Saldırı modülü, isminden de anlaşılacağı üzere, crawler modülü tarafından kendisine gönderilen URL listesindeki adresler üzerinde zafiyet testleri gerçekleştirmektedir. İlgili modül, listede yer alan her bir URL adresine, GET ve POST metotları üzerinden

çeşitli zafiyetleri tetikleyecek zararlı payloadlar göndermektedir. Saldırı modülü tarafından tespit edilen zafiyetlerin kullanıcıya bildirilmesi için geliştirilen modül ise raporlama modülüdür. Aracın geliştirilmesine bağlı olarak, tespit edilen zafiyetler, zafiyetlere neden olan payloadlar, zafiyetin kapatılması için gerekli öneriler kullanıcıya çeşitli dosya formatlarında bildirilmektedir.

Bir web zafiyet tarayıcının sahip olduğu modüller incelendiğinde kapsam belirleme modülünün tam kapsamlı çalışması ve doğru sonuçlar üretmesi, gerçekleştirilecek testin güvenirliği açısından büyük önem taşımaktadır. Çünkü ziyaret edilmemiş bir adreste ya da tespit edilemeyen bir sayfada yer alan bir zafiyet, web uygulamasının saldırganlar tarafından istismar edilmesine neden olacaktır.

4. Web Zafiyet Tarayıcısı- Kapsam Belirleme Modülünün Geliştirilmesi

Bu çalışmada; Sansar isimli, dinamik analiz ve yapay zeka tabanlı bir zafiyet tarayıcının kapsam belirleme modülü işlenmektedir. Söz konusu zafiyet tarayıcının sahip olduğu modüller ve işleyişi Şekil 2' de gösterilmektedir.



Şekil 2. Sansar web uygulama zafiyet tarayıcı modülleri

Geliştirilen web uygulama zafiyet tarayıcı; komut ekranı üzerinden çalışmaktadır. Web uygulama zafiyet tarayıcıya birçok farklı modül ve özelliğin eklenmiş olması, beraberinde farklı amaç ve modlarda çalışma imkânı da sunmaktadır. Şekil 3' de, geliştirilen web uygulama zafiyet tarayıcı içerisinde tanımlanmış argüman listesi gösterilmektedir.

Bu makale çalışması kapsamında işlenen crawler modülü, Şekil 3' de gösterilen argümanlardan –u, –sd, -b, -c, -fs, --username ve –password argümanlarını kullanmaktadır

Şekil 3' de gösterilen argümanlar incelendiğinde, -u ya da –url argümanı ile kapsam belirleme işleminin başlanacağı URL adresi girilmektedir. -c ya da –crawl argümanı, kullanıcı tarafından girilen URL adresinden yola çıkarak tüm sitenin indekslenmesi (crawl edilmesi) için kullanılmaktadır. -sd ya da –subdomain argümanı, verilen URL adresine ait alt domain adreslerinin arama motorları üzerinden toplanması için kullanılmaktadır. -b ya da –brute argümanı, kullanıcı tarafından verilen web uygulaması içerisinde adreslenmemiş, gizli web sayfalarının kaba kuvvet (brute force) tekniği aracılığı ile toplanması için kullanılmaktadır. –username ve –password argümanları ise kullanıcı girişi gereken web uygulamalarının taranması esnasında, başarılı kullanıcı girişi yaparak, login arkasında kalan sayfaların da gezilebilmesi için kullanılmaktadır. -fs ya da –fullscan argümanı ise –c, -sd, -b argümanlarının aynı anda kullanıldığı durumlar için tanımlanmıştır. Geliştirilen uygulama –fs argümanı ile çalıştırıldığında, ilk olarak başlangıç URL adresinden yola çıkarak subdomain adreslerini bulacak, daha sonra kaba kuvvet yöntemi ile gizlenen URL varsa tespit edecek, sonra olarak örümcek modülü ile eldeki URL adreslerini kullanarak tüm web uygulamasını gezecektir. Geliştirilen araç, web uygulama sızma testi kapsamının belirlenmesinde bu üç tekniğin aynı anda kullanıldığı ilk çalışma özelliği taşımaktadır. Şekil 3' de gösterilen argümanlardan –exception argümanı ise zafiyet taraması esnasında, kapsam dışında tutulmak istenen URL adreslerini belirtmek için kullanılmaktadır.

Geliştirilen aracın kullanımının daha iyi anlaşılabilmesi için, http://teztest.com adresinden yola çıkarak gerçekleştirilebilecek farklı türde kapsam belirleme örnekleri aşağıda verilmiştir.

```
parser = argparse.ArgumentParser()
parser.add_argument('-u','--url', dest='url',  required=True, action='store', help= "T
parser.add_argument('-sd', '--subdomain', dest='subdomain', required=False, action='st
parser.add_argument('-c', '--crawl', dest='crawl', required=False, action='store_true'
parser.add_argument('-m', '--method', dest='method', required=False, action='store',he
parser.add_argument('-a', '--attack', dest='attack', required=False, action='store_tru
parser.add_argument('--threading', dest='threading', required=False, action='store_tru
parser.add_argument('-fs', '--fullscan', dest='fullscan', required=False, action='stor
parser.add_argument('--username', dest='username', required=False, action='store',help
parser.add_argument('--password', dest='password', required=False, action='store',help
parser.add_argument('--exception', dest='exception', required=False, action='store',he
parser.add_argument('-b', '--brute', dest='attack', required=False, action='store_true
```

Şekil 3. Web zafiyet tarayıcı kullanım argümanları

- python sansar.py -u http:// teztest.com -c
- python sansar.py -u http:// teztest.com –sd -b
- python sansar.py -u http:// teztest.com -c –username mali –password 123456

Yukarıdaki örnekler incelendiğinde ilk örnekte web zafiyet tarayıcıya -u argümanı ile verilen URL adresinden yola çıkarak tüm siteyi sayfa kaynağından elde ettiği URL adreslerini ve sayfalarda bulunan form girdi alanlarını kullanarak dolaşması söylenmiştir. İkinci kullanım örneğinde, yine aynı URL adresinden yola çıkarak öncelikle alt domain adreslerini belirlemesi, daha sonra da belirlenen adresler üzerinde gizlenmiş URL tespiti için kaba kuvvet saldırısı yapılması söylenmiştir. Üçüncü kullanım örneğinde ise, ikinci kullanım örneğine ek olarak ziyaret ettiği sayfalarda karşılaştığı girdi alanlarına kendisine verilen kullanıcı adı ve parola bilgilerini girmesi söylenmiştir. Böylelikle geliştirilen araç sayfaları dolaşırken bir login sayfası ile karşılaştığında, söz konusu kullanıcı adı ve parola bilgilerini kullanarak oturum açabilecek, kullanıcı giriş sayfası arkasında yer alan gizli sayfaları gezebilecektir.

Geliştirilen web uygulama zafiyet tarayıcı- kapsam belirleme aracının sahip olduğu alt modüller; alt domain bulma modülü, kaba kuvvet modülü ve örümcek modülüdür.

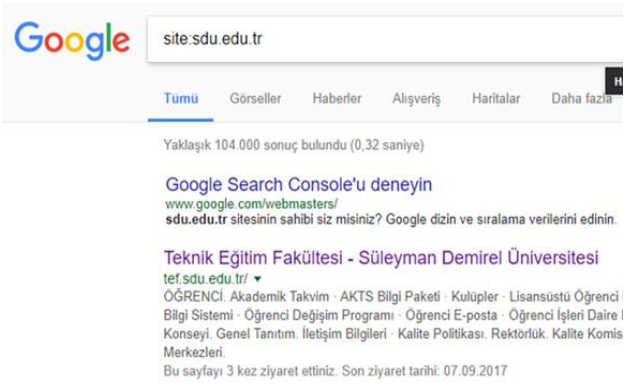
4.1. Alt Domain Bulma Modülü

Günümüzde web uygulamaları üzerinde gerçekleştirilen sızma testlerinde sadece ana domain ve alt domain adreslerinin test edilmesi yaygın bir örnektir. Web uygulamalarının sadece önemli sayfalarının test edilmesi taraftarı olan kullanıcılar, alt domain adreslerinin test kapsamına alınmasını istemektedir. Günümüzde güvenlik uzmanları tarafından alt domain bulma işleminde izlenen çeşitli yöntemler ve kullanılan çeşitli araçlar bulunmaktadır.

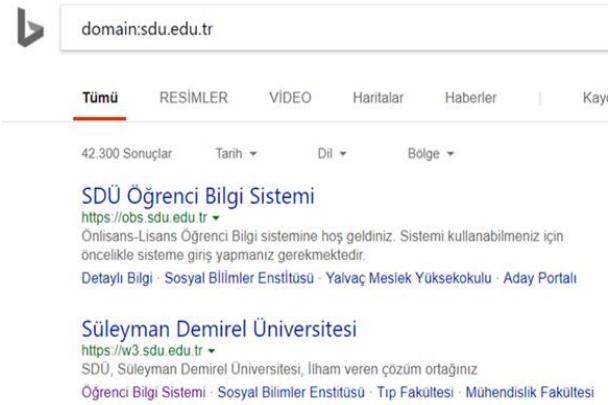
Güvenlik uzmanları tarafından alt domain adreslerinin tespit edilmesinde kullanılan en yaygın yöntem arama motorlarıdır. Google, Baidu, Yahoo, Ask, Bing gibi arama motorları, daha spesifik arama yapılabilmesi için çeşitli gelişmiş arama operatörlerini desteklemektedirler. Google için bu operatörlere Google Dorks adı verilmiştir [27]. Bir alan adı için tanımlanmış tüm alt domain adresleri bulmak amacıyla gerçekleştirilecek Google aramasında “site” operatörü kullanılmalıdır.

Örnek olarak sdu.edu.tr uzantılı alt domain adreslerinin listelenmesi için gerçekleştirilen Google araması Şekil 4’ de şekilde gösterilmektedir.

Bing arama motoru da gelişmiş çeşitli arama operatörlerini desteklemektedir. Geliştirilen kapsam belirleme aracında, bir domain adresine ait alt domainlerin elde edilebilmesi için Bing arama motorunun “domain” operatörü kullanılmıştır. Bing arama motoru içerisinde “domain:sdu.edu.tr” şeklinde arama yapıldığında sdu.edu.tr alt domain adreslerini listelediği görülmüştür. Gerçekleştirilen arama işlemine ait ekran görüntüsü aşağıdaki şekil 5’ de gösterilmiştir.



Şekil 4. Google üzerinden site operatörü ile alt domain bulma işlemi



Şekil 5. Bing üzerinden domain operatörü ile alt domain bulma işlemi

Bing üzerinden alt domain bulma işleminde, geliştirilen araç tarafından ziyaret edilecek URL adresi dinamik olarak oluşturulmakta ve sürekli olarak değiştirilmektedir.

- https://www.bing.com/search?q=domain%3Asdu.edu.tr&first=10

Yukarıdaki URL adresi incelendiğinde search?q=domain%3 ifadesinden sonra alt domain adresleri aranacak domain adresi yer almaktadır. &first= ifadesinden sonra ise arama sonunda dönen yanıt sayfasında, hangi kayıtların gösterileceği bilgisi yer almaktadır. Geliştirilen alt domain bulma modülü, ilgili URL kalıbı içerisine taranacak domain adresini ekleyerek istek yollamakta, isteğe karşılık dönen yanıt içerisinden alt domain adreslerini toplamakta, daha sonra kayıtlar 10’ ar tane listelenecek şekilde, yeni kayıt bulunamaya kadar güncellenmektedir. Geliştirilen alt domain bulma modülünün çalışma altyapısı Tablo 1’ de gösterilmektedir.

Tablo 1. Alt domain bulma modülü çalışma adımları

No	Alt Domain Toplama İşlemi Adımları
1	Alt domainleri toplanacak domain adının alınması
2	Domain adının Bing arama string kalıbı içerisine yerleştirilmesi
3	Oluşturulan dinamik URL adresine istek yapılması ve dönen sayfa kaynağının alınması
4	Sayfa kaynağı içerisindeki alt domain adreslerinin Pyton Beautifulsoup kütüphanesi yardımı ile ayıklanarak kaydedilmesi
5	Oluşturulan dinamik URL adresinin sonraki 10 kaydı gösterecek şekilde güncellenmesi ve tekrar istek atılması
6	İstek atılan yeni URL adresinden dönen sayfa kaynağının alınması ve 4. Adımın tekrar edilmesi
7	5 adımın gösterilecek kaynak kalmayana kadar tekrar edilmesi

Geliştirilen modül kullanılarak gerçekleştirilen tarama ile alt domain adreslerinin elde edilmesinden sonra söz konusu adresler, geliştirilen modelin bulunduğu klasör içerisinde oluşturulmuş olan subDomain_Listesi.txt dosyasına kaydedilmektedir. Şekil 6' da geliştirilmiş olan aracın çalıştırılması ve sdu.edu.tr domainine ait alt domainlerin elde edilmesi işlemi gösterilmektedir.

```
http://www.bing.com/search?q=domain%3asdu.edu.tr&first=1
https://obs.sdu.edu.tr
https://w3.sdu.edu.tr
http://sosyalbilimler.sdu.edu.tr
https://ebys.sdu.edu.tr
http://hastane.sdu.edu.tr
http://fenbilimleri.sdu.edu.tr
http://sgdb.sdu.edu.tr
http://oidb.sdu.edu.tr
https://zigana.sdu.edu.tr
https://bapbs.sdu.edu.tr

http://www.bing.com/search?q=domain%3asdu.edu.tr&first=12
http://yapiisleri.sdu.edu.tr
http://ilahiyat.sdu.edu.tr
https://yosbasvuru.sdu.edu.tr
http://tbmyo.sdu.edu.tr
http://yalvacmyo.sdu.edu.tr
```

Şekil 6. Geliştirilmiş alt domain bulma modülünün çalıştırılması

4.2. Kaba Kuvvet (Brute Force) Modülü

Günümüzde web uygulama geliştiriciler, ziyaretçiler tarafından görülmesini istemedikleri admin paneli ya da robots.txt dosya sayfası gibi sayfalara güvenlik kaygısı ile site içerisinde link vermemektedirler. Uygulama geliştiriciler tarafından gizlenen bu sayfalar saldırganlar için büyük önem ifade etmektedir. Çünkü gizlenmiş admin paneline yapılacak izinsiz bir giriş, tüm site yönetiminin siber saldırganların eline geçmesi anlamına gelmektedir. Oluşabilecek zararın boyutları düşünüldüğünde, söz konusu gizli sayfaların tespit edilebilirliğinin ve güvenliğinin test edilmesi büyük önem taşımaktadır. Gerçekleştirilen çalışma kapsamında geliştirilen kapsam belirleme modülü içerisine, web uygulamasında link verilmemiş gizli sayfaları brute force tekniği ile tespit eden bir modül eklenmiştir. Geliştirilen modül içerisinde yer alan, brute force tekniğinde kaynak olarak kullanılan kelime listesine (wordlist) ait ekran görüntüsü Şekil 7' de gösterilmektedir.

bruteForceCrawler.py | commons.txt

```
22  Admin
23  Administration
24  crm
25  CVS
26  CYBERDOCS
27  CYBERDOCS25
28  CYBERDOCS31
29  INSTALL_admin
30  Log
31  Logs
32  Pages
33  Servlet
34  Servlets
35  SiteServer
36  Sources
37  Statistics
38  Stats
39  W3SVC
40  W3SVC1
41  W3SVC2
42  W3SVC3
43  WEB-INF
44  _admin
45  _pages
46  a
47  aa
48  aaa
49  abc
50  about
51  academic
52  access
53  accessgranted
54  account
55  accounting
56  action
57  actions
58  active
```

Şekil 7. Kaba kuvvet yöntemi ile gizli URL adreslerinin tespit edilmesinde kullanılan kelime listesi

Geliştirilen modülde kaba kuvvet yöntemi ile test edilecek domain ve alt domain adreslerinin sonuna /SANSAR ifadesi eklenmektedir. Daha sonra Şekil 7' de gösterilmekte olan liste içerisinde yer alan uzantılar, SANSAR terimi ile yer değiştirmektedir. Örneğin test edilen alt domain adresi; teztest.com/SANSAR ise ve listede test edilecek sıradaki kelime 'admin' ise test edilecek URL adresi; teztest.com/admin olmaktadır. Şekil 8' de söz konusu işlemin gerçekleştirildiği sınıfa ait kurucu metod gösterilmektedir.

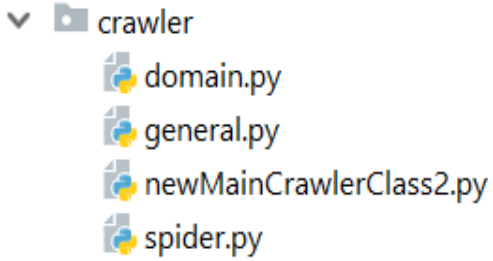
```
class bruteForceModule(Thread):
    def __init__( self,word,url):
        Thread.__init__(self)
        try:
            self.word = word.split("\n")[0]
            self.urly = url.replace('SANSAR',self.word)
            self.url = self.urly
        except Exception as e:
            print(e)
```

Şekil 8. bruteForceModule sınıfı kurucu metodu

Kaba kuvvet yöntemi ile test edilecek URL kalıplarının oluşturulmasından sonra, söz konusu kalıpların geçerli birer adres olup olmadığı test edilmektedir. Bu işlem için Python requests kütüphanesi kullanılmıştır. Gerçekleştirilen işlemde oluşturulan URL kalıplarına GET metodu kullanılarak bağlantı isteğinde bulunulmuştur. Daha sonra, gerçekleştirilen bağlantı isteğinin durum kodu (status code) kontrol edilmiştir. Ziyaret edilen URL adreslerinin geçerli olması durumunda, 200 durum kodu elde edilmelidir. Böylelikle hangi URL adresinin geçerli olup hangisinin geçerli olmadığı tespit edilmektedir.

4.3. Örümcek (Crawler) Modülü

Bir web uygulamasına yönelik olarak sızma testi gerçekleştirileceği zaman ana domain ve alt domainlerin listesi kimi durumlarda test uzmanı için yeterli olmamaktadır. Ana domain ve alt domainler içerisinde bulunmayan bir zafiyet, söz konusu domainler içerisinde yer alan alt URL adreslerinde görülebilmektedir. Kapsam belirleme aracı içerisine eklenen örümcek modülü; kendisine parametre olarak gönderilen URL adresinden başlayarak tüm web sayfasını baştan aşağıya gezmekte, gezdiği URL adreslerini kaydetmektedir. Böylelikle test edilecek web uygulamasının tüm haritası elde edilmektedir. Şekil 9' da crawler modülüne ait sınıflar gösterilmektedir.



Şekil 9. Örümcek modülü sınıfları

Örümcek modülü içerisinde yer alan temel sınıf, newMainCrawlerClass.py sınıfıdır. Bu sınıf; diğer sınıfların çağırılması, toplanan URL adreslerinin text dosyasına kaydedilmesi işlemleri için kullanılmaktadır. Şekil 10' da söz konusu sınıfa ait kurucu metod gösterilmektedir.

```
def __init__(self,url,username,password,contentOfLoginPage,session,exception):
    self.isFinished=False
    self.PROJECT_NAME = 'scanReport'
    self.HOMEPAGE = url
    self.DOMAIN_NAME = get_domain_name(url)
    self.QUEUE_FILE = self.PROJECT_NAME + '/kuyruk.txt'
    self.CRAWLED_FILE = self.PROJECT_NAME + '/bulunan_linkler.txt'
    self.queue = Queue()
    self.general_headers = {'User-Agent': 'Mozilla/5.0 (Windows; U; Windows N
                'Accept-Language': 'en-US;',
                'Accept-Encoding': 'gzip, deflate',
                'Accept': 'text/html,application/xhtml+xml,application/xml;',
                'Connection': 'close'}
    # self.loginUrl = "http://testphp.vulnweb.com/login.php"
    self.session = session
    self.listOfLines=set()
    self.username=username
    self.password=password
    self.finalListOfURLs=set()
    self.contentOfLoginPage= contentOfLoginPage
    self.exception=exception
```

Şekil 10. Örümcek modülü ana sınıfı kurucu metodu

Web uygulamasının crawler modülü tarafından nasıl gezildiğine değinmeden önce parametre olarak aldığı oturum (session) nesnesine değinmek gerekmektedir. Bu çalışma kapsamında geliştirilen web uygulama zafiyet tarayıcı, web sayfalarını gezerken ve kaydedilmiş URL adresleri üzerinde zafiyet testi yaparken aynı oturum nesnesini kullanmaktır. Bunun sebebi, web uygulaması üzerinde söz konusu oturum nesnesi ile bir kez oturum açıldığında, gerçekleştirilecek tüm bağlantıları o oturum üzerinden gerçekleştirmektir.

Günümüzde kullanılan web uygulama tarayıcılar kullanıcı girişi gereken ve giriş sonrası görüntülenebilecek sayfaların gezilmesi ve test edilmesi için kullanıcılardan ilk olarak bir cookie bilgisi istemektedir. Kendilerine verilen cookie ile oturum elde eden araçlar, kapsam belirleme ve test işlemlerini bu oturum üzerinden gerçekleştirmektedir. Fakat bu yöntemin çeşitli dezavantajları bulunmaktadır. Kullanıcılar bir web sayfası üzerinde zafiyet testi yapabilmesi için ilk olarak o sayfayı kendi tarayıcılarından açmalı, kullanıcı girişi yapmalı ve elde ettikleri cookie değerini bir şekilde alarak programa vermelidir. Bu süreç hem uzun hem de acemi kullanıcılar için zor ve hata yapmaya çok müsaittir.

Bu çalışma kapsamında geliştirilen kapsam belirleme aracına, kullanıcı girişinin gerektiği web uygulamalarını test ederken geçerli bir kullanıcı adı ve parola vermek yeterlidir. Geliştirilen araç web uygulamasını gezme esnasında kullanıcı girişi formu ile karşılaştığında, kendisine parametre olarak gönderilen kullanıcı adı ve parola bilgilerini forma girerek başarılı bir şekilde giriş yapmaktadır. Karşılaşılan login sayfası üzerinde kullanıcı girişinin yapılması işleminde Python bünyesindeki Selenium kütüphanesi kullanılmaktadır. Gerçekleştirilen tüm bağlantı işlemleri aynı oturum nesnesi üzerinden gerçekleştirildiği için, bir kez giriş yapıldıktan sonra aynı oturum ile tüm sayfalar başarılı bir şekilde ziyaret edilebilmektedir.

Geliştirilen crawler modülünün bir diğer avantajı, daha önce de bahsedildiği üzere, sayfa üzerinde karşılaştığı formları varsayılan değerler ile doldurmasıdır. Örneğin bir web sayfasında arama formunun tespit edildiği düşünülsün. Sayfa üzerindeki linkleri toplayan fakat formlar üzerinde bir işlem yapmayan bir araç bu formu dikkate almadan işlemine devam edecektir. Fakat bu çalışma kapsamında geliştirilmiş kapsam belirleme aracı söz konusu form ile karşılaştığında, arama inputu içerisine bir değer girmekte ve formu submit etmekte yani göndermektedir. Söz konusu işleme ait adımlar sırası ile şu şekilde gerçekleşecektir; ziyaret edilen web sayfasına ait kaynak kodları elde edildikten sonra, kaynak kodlar içerisindeki <form tagları elde edilmektedir. Form taglarının elde edilmesinden sonra ilk olarak forma ait 'action' değeri bir değişkene atılmaktadır. Bu sayede POST isteği gönderilecek URL adresi elde edilmektedir. Daha sonra form içerisinde yer alan '<input' tagları elde edilmektedir. Böylelikle form bünyesinde yer alan inputlara ait 'name', 'type' gibi değerler kullanılabilmektedir. Form içerisindeki inputlar tespit edildikten sonra, her bir inputa türüne göre tanımlanmış varsayılan değerler atanmaktadır. Söz konusu işlemi gerçekletirilen metoda 'FormScraper' ismi verilmiştir. FormScraper metodu tüm bu işlemlerden sonra crawler modülüne ziyaret etmesi için bir POST nesnesi döndürmektedir. Şekil 11' de formlar içerisinde tespit edilen inputlara, türlerine göre atanacak varsayılan değerler gösterilmektedir.

```
defaults = {
    "checkbox": "default",
    "color": "fffffff",
    "date": "2019-02-02",
    "datetime": "2019-03-19T19:03:34.32",
    "datetime-local": "2019-03-19T19:03",
    "email": "sansar%40mailinator.com",
    "mail": "sansar%40mailinator.com",
    "file": ["pix.gif", "GIF89a", "image/gif"],
    "hidden": "default",
    "month": "2019-03",
    "number": "1337",
    "password": "parolametni",
    "radio": "beton",
    "range": "37",
    "search": "aramametni",
    "submit": "submit",
    "tel": "0555555555",
    "text": "textmetni",
    "time": "19:03",
    "url": "https://akademik.ahievran.edu.tr/sit
    "week": "2019-W03"
}
```

Şekil 11. Formlar içerisinde tespit edilen inputlara atanacak varsayılan değerler

Formscraper metodu tarafından geriye döndürülen POST kalıbı kullanılarak, mevcut session nesnesi aracılığı ile istek gönderilmektedir. Şekil 12' de geliştirilen formscraper aracının kullanımı sonrası üretilen POST metodları gösterilmektedir.

Şekil 12' de gösterilmekte olan web sayfası incelendiğinde; sayfa içerisinde arama işlemi için oluşturulmuş bir formun yer aldığı görülmektedir. FormScrapaer metoduna söz konusu URL adresi parametre olarak verildiğinde, çıktı olarak bir POST kalıbı döndürmüştür. Döndürülen POST kalıbı incelendiğinde, form içerisinde yer alan inputlara ait isimler ve inputların türlerine göre atanan varsayılan değerler görülmektedir.

Şekil 12. FormScrapaer metodu tarafından POST isteklerinin üretilmesi

Geliştirilen kapsam genişletme aracı içerisindeki oturum yönetimi ve otomatik form doldurma özelliklerine değindikten sonra bir web uygulamasının baştan aşağı nasıl gezildiğine değinmek gerekmektedir. Kapsam genişletme aracı içerisindeki ana sınıf, kendisine gönderilen URL, kullanıcı adı, parola gibi bilgileri, gezme modülü içerisindeki bir diğer sınıf olan spider.py sınıfına göndermektedir.

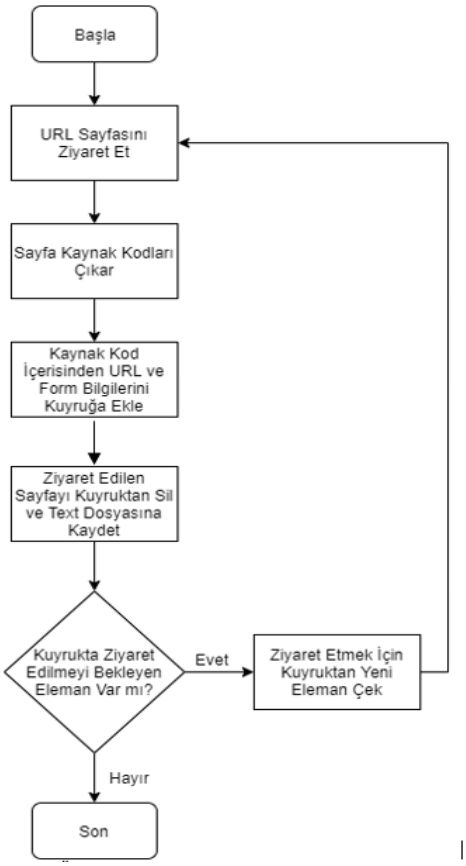
Spider.py sınıfı ilk olarak kendisine gönderilen başlangıç sayfasına bağlanmaktadır. Başlangıç sayfasına gerçekleştirilen bağlantı sonrası gerçekleştirilen ilk işlem, sayfa kaynağını almaktır. Ziyaret edilen sayfaya ait kaynak kodları bir Beautifulsoup nesnesine dönüştürülerek, içerisinde yer alan URL adreslerinin ve form bilgilerinin kazınmasını sağlayan "extractURLsFromSoup" isimli bir metoda gönderilmektedir.

extractURLsFromSoup methodu ilk olarak, kendisine gönderilen sayfa kaynağı içerisinde yer alan URL adreslerini toplamaktadır. Bu işlemde, sayfa kaynağında yer alan "script", "iframe"," a", "frame", "iframe", "button" ve "meta" HTML tagları çıkartılmaktadır. Bu adımdan sonra söz konusu taglar içerisinde yer alan; "src", "href", "action", "hhtp-equiv" alanlarına atanan URL adresleri toplanmaktadır. Toplanan URL adresleri, gezilen sayfa domainine ait olup olmadığı kontrol edildikten sonra ziyaret edilecek URL adresleri listesine eklenmektedir. Bu işlemler sonrasında bir web

sayfası içerisinde verilmiş tüm URL adresleri başarı ile toplanmaktadır.

extractURLsFromSoup tarafından gerçekleştirilen ikinci işlem ise sayfa kaynağını daha önce bahsedilen FormScraper methoduna yollamaktır. FormScraper metodu tarafından, kaynak kodları içerisinden formlar çıkartılıp varsayılan değerler ile doldurulacak ve geriye POST istek kalıpları olarak döndürülecektir. Sayfa kaynağından kazıdığı URL adreslerine, FormScraper metodundan gönderilen POST istek kaloplarını da ekleyen extractURLsFromSoup metodu, elindeki ziyaret edilecekler listesini geriye döndürmektedir.

Spider.py sınıfı, bünyesindeki extractURLsFromSoup metodundan geriye döndürülen, ziyaret edilecek URL ve istek atılacak POST kalıpları listesini kuyruğa eklendikten sonra, bu verilerin elde edildiği URL adresini bir text dosyasına kaydetmektedir. Bu sayede ziyaret edilip kaynak kodu parçalanan bir adresin tekrar ziyaret edilmesi engellenmektedir. Bu işlemden sonra Spider.py sınıfı elindeki kuyruktan bir elemanı çekerek, aynı ziyaret et- kaynak kodu parçala- URL ve POST kalıbı çıkar işlemlerini tekrar etmektedir. Bu işlem döngüsü kuyrukta ziyaret edilecek eleman kalmayana kadar devam etmektedir. Böylelikle bir web uygulaması içerisindeki tüm sayfalar tek tek ziyaret edilecektir. Şekil 13' de bu başlık altında işlenen örümcek modülünün çalışması özetlenmiştir.



Şekil 13. Örümcek modülü çalışma diagramı

5. Bulgular

Bu çalışma kapsamında web uygulama zafiyet tarayıcı- kapsam genişletme modülü geliştirilmiştir. Geliştirilen modül, web güvenliği alanında yaygın olarak kullanılan zafiyet tarayıcıların kapsam genişletme modülleri ve özellikle web uygulamalarındaki URL adreslerinin çıkarılması için geliştirilmiş crawler araçlar ile karşılaştırılmıştır. Yapılan karşılaştırma işlemlerinde kullanılan ilk araç; Metasploit içerisinde yer alan 'msfcrawler' modülü kullanılmıştır. Gerçekleştirilen karşılaştırma işleminde, web uygulama sızma testlerinde yaygın olarak kullanılan, açık kaynak kodlu zafiyet tarayıcılardan WASCAN, Wapiti ve Skipfish kullanılmıştır. Bunun yanında web uygulama sızma testlerinde kapsam belirleme amacıyla yaygın olarak kullanılmakta olan Httrack ve BlackWidow araçları da karşılaştırma işlemine dahil edilmiştir. Gerçekleştirilen işlemde her bir araca 'http://testphp.vulnweb.com' adresi parametre olarak verilmiş ve söz konusu adresten yola çıkarak tüm URL adreslerinin bulunması istenmiştir. Tablo 2' de yapılan test işleminde kullanılan araçlar ve tespit ettikleri URL adres sayıları gösterilmektedir. Gerçekleştirilen tarama işlemleri sonrasında, bu çalışmada geliştirilen web uygulama zafiyet tarayıcı-kapsam belirleme modülü, Httrack aracı ile aynı sayıda ve en çok URL adresini tespit etmeyi başarmıştır. Tablo 2' de gösterilmekte olan veriler incelendiğinde geliştirilen kapsam belirleme modülünün, günümüzde web uygulama sızma testlerinde yaygın olarak kullanılan, açık kaynak kodlu WASCAN, Wapiti ve Skipfish test araçlarının kapsam belirleme modüllerinden daha fazla URL adresi topladığı tespit edilmiştir.

Tablo 2. Sayfa dolaşma için kullanılan araçlar ve tespit edilen URL address sayıları

Kullanılan Araç	URL Adres Sayısı
Sansar ve Httrack	75
Metasploit (msfcrawler)	66
Wapiti	57
Skipfish	50
BlackWidow	40
Wascan	12

Şekil 14' de geliştirilen kapsam belirleme modülünün 'http://testphp.vulnweb.com' adresi üzerinde yaptığı URL toplama işlemine ait ekran görüntüsü gösterilmektedir.

6. Tartışma ve Sonuç

Web uygulamalarını, gerçek bir siber saldırı öncesinde tam olarak güvenli hale getirmek için kullanılan en yaygın yöntem sızma testleridir. Dinamik analiz tabanlı web uygulama zafiyet tarayıcılar, web uygulama sızma testlerini hızlı ve güvenilir gerçekleştirdikleri için sıklıkla tercih edilmektedirler. Web uygulama zafiyet tarayıcılar

Şekil 14. 'http://testphp.vulnweb.com' adresi üzerinde gerçekleştirilen kapsam belirleme işlemi

temel olarak kapsam belirleme, saldırı ve raporlama modülleri olmak üzere 3 kısımdan oluşmaktadır. Kapsam belirleme modülü, taranacak URL adresleri ve formların belirlenmesinde kullanılmaktadır. Bir web uygulamasının tam olarak sızma testine tabi tutulabilmesi için, sahip olduğu tüm sayfalar eksiksiz belirlenerek, saldırı modülüne verilmelidir. Bu çalışmada web uygulamaları içerisinde yer alan sayfaları belirlemede 3 farklı yöntem kullanan bir kapsam genişletme aracı geliştirilmiştir. Geliştirilen araç, arama motorları, kaba kuvvet yöntemi ve crawler aracını kullanarak web uygulaması içerisinde yer alan URL adreslerini belirlemektedir. Önerilen metodoloji ve geliştirilen araç, literatürde 3 farklı yöntemin ilgili problem üzerinde kullanıldığı ilk çalışma olma özelliği taşımaktadır. Yapılan test işlemlerinde, geliştirilen araç, literatürde kapsam belirleme amacı ile kullanılan 5 farklı araç ile karşılaştırılmış ve en fazla URL adresi tespit etmeyi başaran araç olmuştur. Bu çalışmada geliştirilen kapsam genişletme modülü, web uygulama sızma testlerinde aktif olarak kullanılmaya hazır durumdadır.

Kaynakça

[1] Andreu A. 2006, Professional Pen Testing For Web Applications. John Wiley and Sons, 9-10.

[2] Yalçınkaya, M. A. 2020. Yapay Zeka ve Dinamik Analiz Tabanlı Web Uygulama Zafiyet Tarayıcısı. Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, 196s, Isparta.

[3] Vieira, T., Serrão, C. 2016. Web security in the finance sector. 11th International Conference for Internet Technology and Secured Transactions (ICITST), 5-7 December, Barcelona, Spain.

[4] Akrout, R., Alata, E., Kaaniche, M., Nicomette, V. 2014. An automated black box approach for web vulnerability identification and attack scenario generation. Journal of the Brazilian Computer Society, 20(1), 4.

[5] Seng, L. K., Ithnin, N., Said, S. Z. M. 2018. The approaches to quantify web application security scanners quality: a review. International Journal of Advanced Computer Research, 8(38), 285-312.

[6] Idrissi, S. E., Berbiche, N., Guerouate, F., Shibi, M. 2017. Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. International Journal of Applied Engineering Research, 12(21), 11068-11076.

[7] Esposito, D., Rennhard, M., Ruf, L., Wagner, A. 2018. Exploiting the potential of web application vulnerability scanning. The Thirteenth International Conference on Internet Monitoring and Protection, 22-26 July, Barcelona, Spain.

[8] Mantra, I. G. N., Alaydrus, M., Misni, H. M. 2016. The web security and vulnerability analysis model on Indonesia Higher Education institution. International Conference on Informatics and Computing, 28- 30 October, Mataram,Indonesia.

[9] Esposito, D., Rennhard, M., Ruf, L., Wagner, A. 2018. Exploiting the potential of web application vulnerability scanning. International Conference on Internet Monitoring and Protection, 22-26 July, Barcelona, Spain.

[10] Akrout, R., Alata, E., Kaaniche, M., Nicomette, V. 2014. An automated black box approach for web

vulnerability identification and attack scenario generation. Journal of the Brazilian Computer Society, 20(1), 4.

[11] Mutai, H. 2019. Hybrid Multi-Agents System Vulnerability Scanner For Detecting SQL Injection Attacks In Web Applications. PhD Thesis. University of Nairobi, 73p, Nairobi.

[12] Mukhopadhyay, I., Goswami, S., Mandal, E. 2014. Web penetration testing using nessus and metasploit tool. IOSR Journal of Computer Engineering, 16(3), 126-129.

[13] Dessiatnikoff, A., Akrout, R., Alata, E., Kaâniche, M., Nicomette, V. 2011. A clustering approach for web vulnerabilities detection. Pacific Rim International Symposium on Dependable Computing, 12-14 December, California, USA.

[14] Munoz, F. R., Villalba, L. G. 2013. Methods to Test Web Applications Scanners. Conference on Information Technology. 11- 14 November, Islamabad, Pakistan.

[15] Rapid7, 2019. Metasploit Framework. https://www.metasploit.com/ (Erişim Tarihi: 11.11.2019).

[16] Roche, X. 2012. Httrack Website Copier. https://www.httrack.com/ (Erişim Tarihi: 10.11.2019)

[17] XeroSecurity, 2018. Black Widow. https://github.com/1N3/BlackWidow (Erişim Tarihi: 12.11.2019)

[18] Zalewski, M., Heinen, N., Roschke, S. 2011. Skipfish-Web Application Security Scanner. https://code.google.com/archive/p/skipfish/ (Erişim Tarihi: 13.11.2019)

[19] Surribas, N. 2006. Wapiti Web-Application Vulnerability Scanner. http://wapiti.sourceforge.net/ (Erişim Tarihi: 13.11.2019)

[20] Wascan, 2018. Web Application Vulnerability Scanner. https://github.com/m4ll0k/WAScan (Erişim Tarihi: 13.11.2019)

[21] Burlu, K. 2010. Bilişimin Karanlık Yüzü, Nirvana Yayınları, 479s, Ankara.

[22] Yalçınkaya, M. A. 2015. Gelişmiş Hedef Odaklı Siber Saldırılar, Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 186s, Isparta.

[23] Jimenez, R. E. L. 2016. Pentesting on Web Applications using Ethical Hacking. IEEE 36th Central American and Panama Convention, 9-11 November, San Jose, Costa Rica.

[24] Sarıman G. 2015. Yazılım Güvenliği Test Ve Değerlendirme Aracı Geliştirilmesi. Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, 141s, Isparta

[25] Doupé, A., Cavedon, L., Kruegel, C., Vigna, G. 2012. Enemy Of The State: A State-Aware Black-Box Web Vulnerability Scanner. 21st USENIX Security Symposium, 8-10 August, Bellevue, USA.

[26] Doupé, A., Cova, M., Vigna, G. 2010. Why Johnny Can't Pentest: An Analysis Of Black-Box Web Vulnerability Scanners. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Germany.

[27] Hudak, P. 2017. The Art of Subdomain Enumeration. https://blog.sweepatic.com/art-of-subdomain-enumeration/ (Erişim Tarihi: 20.10.2019)