Web Uygulamalarına Yönelik Siber Saldırıların Log Analizi ile Tespiti

Yağız BİLGİLİ (231307120)
Bilişim Sistemleri Mühendisliği Bölümü
Teknoloji Fakültesi
Kocaeli, İzmit
yagizbilgili0@gmail.com

Soner ÇELİK (231307118)
Bilişim Sistemleri Mühendisliği Bölümü
Teknoloji Fakültesi
Kocaeli, İzmit
soner.41.2000@gmail.com

***Abstract*—**Bu çalışmada, web uygulamalarına yönelik siber saldırıların HTTP log analizi ve makine öğrenmesi teknikleri kullanılarak tespiti incelenmektedir. Yöntem, Acunetix, Netsparker ve OWASP ZAP gibi popüler saldırı simülasyon araçlarını kullanarak saldırı logları oluşturmayı amaçlamaktadır. Ayrıca, Reddit ve Wikipedia gibi platformlarda normal kullanıcı davranışları simüle edilerek temiz log verileri elde edilmiştir. Makine öğrenmesi algoritmaları, saldırı loglarını normal kullanıcı loglarından ayırmak ve sınıflandırmak için uygulanmıştır. Sonuçlar, log tabanlı analizlerin siber tehditlerin erken tespiti için etkili ve ölçeklenebilir bir çözüm sunduğunu göstermektedir.

Keywords*—Siber saldırılar, web uygulamaları, HTTP log analizi, makine öğrenmesi, saldırı tespiti, sınıflandırma algoritmaları, zafiyet tarama araçları.*

I. GİRİŞ

Günümüz dijital dünyasında web uygulamaları, bireylerin ve kurumların en önemli veri paylaşım araçlarından biri haline gelmiştir. Ancak, bu uygulamalara yönelik siber saldırılar, veri güvenliği ve bütünlüğü açısından ciddi tehditler oluşturmaktadır. Siber saldırıların yaygınlaşması, özellikle kritik öneme sahip sistemlerin güvenlik gereksinimlerini artırmış ve bu saldırıların tespitine yönelik etkili yöntemlerin geliştirilmesini zorunlu hale getirmiştir.

Web uygulamalarına yönelik siber saldırıların büyük bir kısmı, istemcilerden sunuculara gönderilen HTTP isteklerinin içerik ve yapısında anormallikler meydana getirir. Bu anormallikler, genellikle log dosyalarında sıra dışı veya beklenmeyen davranışlar olarak izlenebilir. Örneğin, SQL enjeksiyonu, XSS (Cross-Site Scripting) ve uzaktan dosya dahil etme gibi saldırılar, loglarda belirli bir deseni takip eder ve sistemlerin güvenlik açığından yararlanmayı amaçlar. Bu çalışmada, HTTP loglarını analiz ederek siber saldırıları tespit etmek amacıyla makine öğrenmesi teknikleri kullanılmaktadır. Log analizi, anormalliklerin belirlenmesi ve saldırı desenlerinin ortaya çıkarılması açısından kritik bir rol oynar.

Makine öğrenmesi, veri kümelerindeki belirli saldırı modellerini tanımak ve bu modelleri kullanarak yeni verilerde anormallikleri tespit etmek için güçlü bir yöntemdir. Büyük veri kümeleri, özellikle siber güvenlik gibi yüksek hacimli log verisi içeren alanlarda makine öğrenmesi algoritmalarının başarıyla uygulanmasını sağlar. Bu projede, Acunetix, Netsparker ve OWASP ZAP gibi yaygın kullanılan saldırı simülasyon araçları ile bir web uygulamasına gerçek saldırılar düzenlenmiş ve bu saldırılar sonucunda oluşan HTTP logları kaydedilmiştir. Bu loglar, saldırı içeren veriyi temsil etmekte ve potansiyel bir tehdit durumunda sistemin verdiği tepkileri anlamamızı sağlamaktadır.

Saldırı içermeyen (temiz) log verilerini elde etmek amacıyla, Reddit ve Wikipedia gibi güvenilir web platformlarında normal kullanıcı davranışları sergilenmiş ve bu sırada oluşan HTTP logları bir proxy sunucu aracılığıyla kaydedilmiştir. Böylece, saldırı ve temiz log verileri ile oluşturulan veri seti üzerinde çeşitli sınıflandırma algoritmaları uygulanarak, sistemin saldırı içerikli ve temiz logları ayırt etme yeteneği test edilmiştir. Bu analiz sayesinde, makine öğrenmesi tabanlı modellerin saldırıları tanımlama yeteneği ölçülmüş ve gelecekte bu sistemlerin daha fazla gelişmesi hedeflenmiştir. Bu bağlamda, projenin temel amacı, bir web uygulamasına yönelik olası saldırıların log verileri üzerinden tespit edilmesini sağlayan bir model geliştirmektir. Çalışmada hem saldırı hem de temiz logların incelenmesi, doğru sınıflandırma yapılarak potansiyel tehditlerin hızlı ve etkili bir şekilde belirlenmesini amaçlamaktadır.

II. LİTERATÜR TARAMASI

Siber saldırılar, özellikle web uygulamaları üzerinde büyük tehditler oluşturmakta ve bu saldırıların erken tespiti, sistem güvenliğini sağlamak adına büyük önem taşımaktadır. Web uygulamalarına yönelik saldırılar, genellikle uygulamaların zafiyetlerinden yararlanarak veri sızıntılarına veya sistem manipülasyonlarına yol açmaktadır. Bu nedenle, saldırıların tespit edilmesi için çeşitli yöntemler geliştirilmiştir.

A. Siber Saldırı Tespiti ve Log Analizi

Web uygulamalarına yönelik siber saldırıların tespit edilmesinde, genellikle sistem logları önemli bir rol oynamaktadır. Log dosyaları, saldırıların izlerini taşıyan ve sistemin durumunu gösteren önemli veriler sunar. Bununla birlikte, bu logların manuel olarak incelenmesi zaman alıcı ve hataya açıktır.

B. Web Güvenliği ve Saldırı Araçları

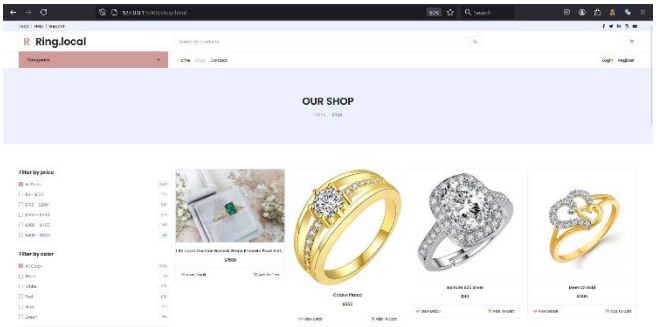
Web uygulamalarına yönelik saldırıların tespiti ve önlenmesi konusunda çeşitli araçlar geliştirilmiştir. Acunetix, Netsparker ve OWASP ZAP gibi araçlar, güvenlik taramaları yaparak web uygulamalarındaki zafiyetleri tespit etmeyi amaçlar. Bu araçlar, özellikle SQL enjeksiyonu, XSS gibi yaygın saldırıları simüle etmek için kullanılır.

III. YÖNTEM

Bu çalışmada, web uygulamalarına yönelik siber saldırıları tespit etmek amacıyla Flask tabanlı bir web uygulaması geliştirilmiş ve bu uygulama üzerinde çeşitli siber saldırılar simüle edilmiştir. Uygulama, günümüz e-

ticaret platformlarının özelliklerini taşıyan, Python-Flask-MySQL ile geliştirilmiş basit bir sistemdir. Bu sistemin temel amacı, saldırıların loglarını toplayarak, bu logların analiz edilmesi için bir veri kümesi oluşturmaktır.

Geliştirilen web uygulaması aşağıdaki gibidir:



A. *Veri Toplama*

Veri toplama aşamasında, web uygulamasına yapılan saldırılar ve normal kullanıcı aktiviteleri üzerinden log verileri elde edilmiştir. Bu süreç, Acunetix, Netsparker ve OWASP ZAP gibi siber saldırı araçları kullanılarak gerçekleştirilmiştir. Bu araçlar aracılığıyla, uygulamaya çeşitli saldırılar (örneğin SQL enjeksiyonu, XSS) gerçekleştirilmiş ve saldırılara ait HTTP istekleri loglanmıştır.

Normal kullanıcı aktiviteleri ise, Wikipedia ve Reddit gibi platformlarda yapılan sıradan gezinme işlemleriyle elde edilmiştir. Bu aktiviteler, gerçek kullanıcı etkileşimlerini simüle etmek amacıyla bir proxy aracı kullanılarak kaydedilmiştir.

Log verilerinin toplanması için geliştirilmiş olan Flask uygulamasına entegre edilen bir middleware kullanılmıştır. Her gelen HTTP isteği, bu middleware tarafından işlenir ve loglanır. Her bir HTTP isteği, istek tipi (GET, POST, vb.) bazında ayrı ayrı bir dizine kaydedilir. Loglar, istekle ilgili temel bilgileri (metod, URL, başlıklar, veri) içerir ve bu loglar, saldırı içeren isteklerin tespitinde kullanılacak veri setini oluşturur.

Örnek bir saldırı logu şu şekilde olabilmektedir:

```
{
    "method": "GET",
    "url": "/?p=%24%20%3CScRipT%205-0%2A3%289/1%3D%3Ealert%281%29%3C/ScRipT%20giveanswerhere%3D%3F------",
    "headers": {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0",
        "Connection": "close"
    },
    "isAttack":1,
    "data": ""
},
```

Bu log, yapılan isteklerin HTTP metodunu, URL'yi, başlıkları ve istekle birlikte gönderilen veriyi içerir. “isAttack” etiketi, bu isteğin bir saldırı içerip içermediğini belirtir. Saldırı istekleri, analiz aşamasında tespit edilmek üzere işaretlenmiştir.

B. *Middleware ve Loglama Mekanizması*

Flask uygulamasında, her gelen HTTP isteği bir middleware aracılığıyla işlenmektedir. Bu middleware, before_request fonksiyonu ile HTTP isteklerini işler ve gelen her istek için ayrı bir log kaydeder. Loglar, HTTP metoduna göre farklı alt dizinlere kaydedilir. Örneğin, GET istekleri “/logs/GET/” dizinine, POST istekleri ise “/logs/POST/” dizinine kaydedilecektir.

Her bir istek için, dosya adı benzersiz bir şekilde oluşturulur ve log verileri JSON formatında saklanır. Bu loglar, daha sonraki adımlarda saldırıların tespit edilmesi için analiz edilmek üzere veri seti olarak kullanılacaktır.

Bu aşamada geliştirilen loglama mekanizması aşağıdaki gibidir:

```python
LOG_DIR = "logs"

def log_request(req):
    method = req.method
    method_dir = os.path.join(LOG_DIR, method)

    if not os.path.exists(method_dir):
        os.makedirs(method_dir)

    log_files = os.listdir(method_dir)
    log_number = len(log_files) + 1
    log_file = os.path.join(method_dir, f"{log_number}.json")

    log_data = {
        "method": method,
        "url": req.path,
        "headers": dict(req.headers),
        "data": req.data.decode('utf-8') if req.data else '',
        "timestamp": datetime.now().isoformat()
    }

    with open(log_file, "w") as f:
        json.dump(log_data, f, indent=4)

# Middleware
@myApp.before_request
def before_request_logging():
    log_request(request)

@myApp.after_request
def after_request(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type,Authorization')
    response.headers.add('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS')
    return response
```

C. *Toplanan Verilerin Hazırlığı*

Veri toplama aşamasının sonunda, elde edilen HTTP logları, saldırı içeren ve normal kullanıcı isteklerini ayırt edebilmek için işaretlenmiş bir veri kümesi oluşturulacaktır. Bu veriler, ilerleyen aşamalarda makine öğrenmesi teknikleri kullanılarak incelenmek üzere işlenmeye hazır hale getirilecektir. Şu anda elde edilen veriler, sadece logların toplanması ve ön işlenmesi adımlarını içermektedir.

IV. GELECEK ADIMLAR VE PROJE İLERLEYİŞİ

Veri toplama aşamasının tamamlanmasının ardından, elde edilen log verileri üzerinde yapılacak olan saldırı tespiti için makine öğrenmesi ve sınıflandırma algoritmalarını kullanma aşamasına geçilecektir. Bu aşama, siber saldırıların tespiti ve sınıflandırılması için temel olacak verileri hazırlamayı amaçlamaktadır. Bu bölümde, ilerleyen adımlarda yapılacak işlemler özetlenmiştir:

A. *Verilerin İşlenmesi ve Özellik Çıkartımı*

Toplanan HTTP logları, verilerin analizine uygun hale getirilmek üzere işlenecektir. Bu aşamada, aşağıdaki işlemler gerçekleştirilecektir:

- **Veri Temizleme**: Bozuk, eksik veya hatalı veriler ayıklanacak ve verilerin doğru bir şekilde analiz edilmesi sağlanacaktır.
- **Özellik Çıkartımı (Feature Extraction)**: HTTP isteklerinin içerdiği önemli öznitelikler (method, url, headers, vb.) belirlenip çıkartılacaktır. Bu öznitelikler, saldırı tespiti için modelin eğitilmesinde kullanılacaktır.
- **Etiketleme**: Saldırı istekleri **“isAttack”: 1** şeklinde etiketlenmiş olup, bu etiketler modelin doğru bir şekilde eğitilmesi için kullanılacaktır.

B. *Makine Öğrenmesi Modellerinin Eğitilmesi*

Özellik çıkartımı aşamasından sonra, elde edilen veriler üzerinde makine öğrenmesi algoritmaları ile saldırı tespiti için modeller eğitilecektir.

V. Makine Öğrenmesi Projelerinde Web Kazıma İşleminin Önemi

Makine öğrenmesi projelerinde, veri toplama ve veri işleme aşamaları, modelin başarısını doğrudan etkileyen kritik adımlardır. Web kazıma (web Scraping), özellikle internet üzerinden veri toplama gereksinimi duyulan projelerde önemli bir teknik olarak öne çıkmaktadır. Web kazıma, bir web sitesinin içeriklerini otomatik olarak toplamak için kullanılan bir tekniktir ve genellikle HTML sayfalarındaki verileri sistematik bir şekilde çıkarma işlemi olarak tanımlanır.

Makine öğrenmesi projelerinde web kazıma, çeşitli veri kümelerinin toplanmasında ve analiz edilmesinde önemli bir rol oynar. Özellikle çevrimiçi ortamda bulunan büyük miktarda veriyi derlemek ve işlemek, makine öğrenmesi modellerinin eğitilmesinde ihtiyaç duyulan kaliteli veriye ulaşmak için kritik öneme sahiptir. Bu bağlamda, web kazıma işlemiyle elde edilen veriler, analizler ve tahminler için temel veri kaynağını oluşturabilir.

VI. Demo: E-Ticaret Sitesinden Web Kazıma Yöntemi İle Laptop Fiyat ve Özellik Verisi Çekme

Bu bölümde, makine öğrenmesi projelerinde veri toplama sürecine dair bir demo örneği sunulmaktadır. E-ticaret sitesinden laptop fiyatları ve özelliklerinin çekilmesi, veri toplama sürecinin önemini vurgulamak amacıyla gerçekleştirilmiştir. Bu işlem, makine öğrenmesi algoritmalarının geliştirilmesinde kritik bir adım olan veri elde etme sürecini anlamak için gereklidir. Özellikle, sınıflandırma ve model eğitiminde kullanılan büyük veri kümelerinin oluşturulabilmesi için web kazıma tekniklerinin etkinliği gösterilmiştir.

Bu çalışma, projeye özgü olmasa da, makine öğrenmesi süreçlerine doğrudan katkı sağlamakta ve veri toplamanın bu süreçlerde nasıl bir rol oynadığını ortaya koymaktadır.

A. *Kullanılan Kütüphaneler ve Araçlar*

- **requests**: HTTP istekleri yaparak web sayfalarının içeriğini almak için kullanılır.
- **BeautifulSoup (bs4)**: Web sayfasının HTML içeriğini analiz etmek ve istenilen veriyi çekmek için kullanılır.
- **selenium**: Dinamik içerik yükleyen sayfalardan veri çekmek için kullanılır. Bu sayede JavaScript ile oluşturulan içerikler de çekilebilir.
- **webdriver-manager**: WebDriver için gereken sürücülerin otomatik olarak indirilmesini sağlar.
- **time**: Web sayfası yüklenirken bir süre beklemek için kullanılır.

B. *Tarayıcıyı Gizli Modda Çalıştırma*

Bu kısım, web kazıma işleminin arka planda (görünmeyen bir tarayıcıda) çalışmasını sağlar. Kullanıcıyı rahatsız etmeden, ekran açılmadan işlem yapılır.

```
# Tarayıcıyı gizli modda çalıştırma
options = Options()
options.headless = True  # Tarayıcıyı gizli modda çalıştır
```

C. *WebDriver Başlatma*

Bu satır, tarayıcıyı başlatmak için gerekli olan WebDriver'ı yükler ve Chrome tarayıcısını gizli modda başlatır.

```
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
```

D. *Sayfalardan Ürün Linklerini Toplama*

Bu fonksiyon, belirtilen sayfalarda bulunan tüm ürün linklerini toplar.

```
def get_all_links():
    """
    Verilen sayfalarda bulunan tüm ürün linklerini alır.
    """
    all_links = []

    for page_num in pages:
        url = f"{base_url}{page_num}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        product_bodies = soup.find_all('div', class_='product-body')

        for product in product_bodies:
            title_link = product.find('a', class_='title')
            if title_link:
                href = title_link.get('href')
                all_links.append(href)

    return all_links
```

Her bir sayfa için:

- **requests** ile sayfanın HTML içeriği alınır.
- **BeautifulSoup** ile HTML parse edilir.
- Ürünlerin bulunduğu HTML elementlerinden linkler çekilir ve bir listeye eklenir.

E. *Ürün Verilerini Çekme*

Bu fonksiyon, belirli bir ürün linkine giderek o ürünün detaylı verilerini toplar:

```python
def get_product_data(link):
    """
    Verilen linkteki ürün verilerini çeker.
    """
    driver.get(f'https://www.itopya.com{link}')
    time.sleep(3)


    soup = BeautifulSoup(driver.page_source, 'html.parser')


    table = soup.find('table', {'class': 'table table-product-detail'})

    product_data = {}


    if table:
        rows = table.find_all('tr')
        for row in rows[1:]:
            columns = row.find_all('td')
            if len(columns) > 1:
                key = columns[0].get_text(strip=True)
                value = columns[1].get_text(strip=True)
                product_data[key] = value

    return product_data
```

- selenium kullanılarak sayfa yüklenir (JavaScript ile yüklenen veriler için gereklidir).
- Sayfa yüklendikten sonra BeautifulSoup ile HTML içeriği çekilir ve ürünün özelliklerini içeren tablo bulunur.
- Tabloyu analiz ederek her bir özelliği (örneğin, RAM, işlemci türü, fiyat) anahtar-değer çiftleri olarak product_data sözlüğüne ekler.belirtilen sayfalarda bulunan tüm ürün linklerini toplar.

F. *Verilerin JSON Formatında Kaydedilmesi*

Bu fonksiyon, toplanan tüm ürün verilerini JSON formatında bir dosyaya kaydeder. Bu sayede veriler düzenli ve erişilebilir bir biçimde saklanır.

```python
def save_to_json(data, filename='all_product_data.json'):

    with open(filename, mode='w', encoding='utf-8') as json_file:
        json.dump(data, json_file, ensure_ascii=False, indent=4)
    print(f"Tüm ürün verileri '{filename}' dosyasına kaydedildi.")
```

G. *Ana Fonksiyon (Main)*

Bu fonksiyon, programın ana işlevselliğini yönetir:

```python
def main():

    print("Ürün linkleri toplanıyor...")
    all_links = get_all_links()


    all_products_data = []


    print("Ürün verileri toplanıyor...")
    for link in all_links:
        product_data = get_product_data(link)
        if product_data:
            all_products_data.append({
                "product_link": f'https://www.itopya.com{link}',
                "product_data": product_data
            })
            print(f"{link} için veri toplandı.")
        else:
            print(f"'{link}' sayfasında 'table table-product-detail' class'ına sahip tablo bulunamadı.")


    save_to_json(all_products_data)


if __name__ == "__main__":
    main()


    driver.quit()
```

- **get_all_links()** ile ürün linklerini toplar.
- Her bir ürün linki için get_product_data() fonksiyonunu çağırarak ürün bilgilerini toplar.
- Verileri save_to_json() ile JSON dosyasına kaydeder.

REFERENCES

[1] *Acunetix, 2024. [Online]. Available: https://www.acunetix.com/*

[2] *OWASP Foundation, "OWASP Top Ten," OWASP, 2024. [Online]. Available: https://owasp.org/www-project-top-ten/*

[3] *Netsparker, "Official Documentation," Netsparker, 2024. [Online]. Available: https://www.netsparker.com/*

[4] *OWASP Foundation, "OWASP ZAP - Zed Attack Proxy," OWASP, 2024. [Online]. Available: https://www.zaproxy.org/*

[5] *Armin Ronacher, "Flask Documentation," Flask, 2024. [Online]. Available: https://flask.palletsprojects.com/*

[6] *Oracle Corporation, "MySQL Documentation," MySQL, 2024. [Online]. Available: https://dev.mysql.com/doc/*

[7] *Selenium, "Selenium Documentation," Selenium, 2024. [Online]. Available: https://www.selenium.dev/documentation/en/*

[8] *Leonard Richardson, "BeautifulSoup Documentation," BeautifulSoup, 2024. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/*

[9] *GitHub, "Webdriver Manager for Python," GitHub, 2024. [Online]. Available: https://github.com/SergeyPirogov/webdriver_manager*

[10] *Wikipedia, "Cross-Site Scripting (XSS)," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Cross-site_scripting*

[11] *Wikipedia, "SQL Injection," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/SQL_injection*

[12] *Wikipedia, "Web Scraping," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Web_scraping*

[13] *Web Scraping Tutorials, "A Guide to Web Scraping with Python," Real Python, 2024. [Online]. Available: https://realpython.com/beautiful-soup-web-scraper-python/*

[14] *Wikipedia, "HTTP," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/HTTP*

[15] *Web Security Academy, "Web Application Security," PortSwigger, 2024. [Online]. Available: https://portswigger.net/web-security*

[16] *PortSwigger, "Burp Suite Documentation," PortSwigger, 2024. [Online]. Available: https://portswigger.net/burp*

[17] *Wikipedia, "Cryptography," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Cryptography*

[18] *Python Software Foundation, "Python Documentation," Python, 2024. [Online]. Available: https://docs.python.org/3/*

[19] *Stack Overflow, "Machine Learning Basics," Stack Overflow, 2024. [Online]. Available: https://stackoverflow.com/questions/tagged/machine-learning*

[20] *Wikipedia, "Machine Learning," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning*