ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ

MÜHENDİSLİK MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2023-2024 BAHAR DÖNEMİ

BİÇİMSEL DİLLER VE OTOMATA

DERSİ PROJE FİNAL RAPORU

OTOMATLAR KULLANARAK "DoS" SALDIRI TESPİTİ

Öğretim Üyesi:

Prof. Dr. Ahmet YAZICI

Hazırlayan:
BD_grpA_Grup_15

Umut ÖZTÜRK – 152120211052

Abdullah Taha AYDIN – 152120211055

Atakan BERBER – 152120211057

Haziran 2024

İçindekiler

GİRİŞ

DoS(Denial-of-service Attack)

Hizmet Dışı Bırakma (DoS) saldırıları, bir ağın, sunucunun veya web sitesinin meşru kullanıcı trafiğini engelleyerek hizmet veremez hale gelmesini amaçlayan kötü niyetli girişimlerdir. Bu saldırılar, genellikle hedeflenen sistemi aşırı yük altında bırakarak veya kaynaklarını tüketerek gerçekleştirilir. Daha sofistike bir versiyonu olan Dağıtık Hizmet Dışı Bırakma (DDoS) saldırıları, birden fazla kaynaktan saldırıyı yürüterek tespit ve engellenmelerini daha zor hale getirir.

DDoS(Distributed Denial-of-Service Attack)

Dağıtık Hizmet Dışı Bırakma (DDoS) saldırıları, birden fazla kötü niyetli kaynaktan gelen aşırı trafik ile bir hedefin internet hizmetlerini bozmayı amaçlayan siber saldırılardır. DDoS saldırıları, tek bir noktadan yapılan Hizmet Dışı Bırakma (DoS) saldırılarının aksine, genellikle botnetler olarak adlandırılan, kontrol altına alınmış çok sayıda bulaşmış bilgisayar kullanılarak gerçekleştirilir. Bu yöntem, saldırının kaynağını tespit etmeyi ve savunmayı önemli ölçüde zorlaştırır. DDoS saldırıları, geniş bant genişliğini tüketme, sunucu kaynaklarını aşırı yüklenme veya ağ altyapısını bozma yoluyla hizmetleri kullanılamaz hale getirir. Bu saldırılar, finansal sektör, e-ticaret siteleri, online oyunlar ve hükümet siteleri gibi çeşitli sektörlerdeki kuruluşları hedef alabilir ve ciddi ekonomik zararlara, itibar kaybına ve kullanıcıların güvenini sarsmaya yol açabilir.

DDoS saldırılarına karşı savunma stratejileri, ağ güvenliği uzmanları tarafından sürekli olarak geliştirilmekte ve uygulanmaktadır. Bu stratejiler, trafik analizi, davranış tabanlı bloklama, saldırı imzalarının tespiti ve dağıtık savunma sistemleri gibi çeşitli teknikler içerir. DDoS saldırılarının çeşitliliği ve evrimi, savunma mekanizmalarının da sürekli olarak güncellenmesini gerektirmektedir.

DoS ve DDoS'un Tarihçesi

DoS saldırılarının tarihçesi, internetin erken günlerine kadar uzanır. 1990'ların ortalarında, internetin yaygınlaşmasıyla birlikte, bu tür saldırılar daha görünür ve yıkıcı hale geldi. 1996'da ünlü bir olayda, Panix, New York City merkezli bir internet servis sağlayıcısı, DoS saldırısına uğrayan ilk büyük kurbanlardan biri oldu. 2000'lerin başında, büyük web sitelerine yönelik DDoS saldırıları, bu tehdidin ciddiyetini ve potansiyel zararını tüm dünyaya gösterdi.

DoS ve DDoS'un Siber Güvenlik Alanındaki Yeri

DoS ve DDoS saldırıları, siber güvenlik alanında önemli bir odak noktasıdır. Bu saldırılar, işletmeler, hükümetler ve bireyler için ciddi tehditler oluşturarak, hizmetlerin kesintiye uğramasına, itibar kaybına ve önemli mali zararlara yol açabilir. Güvenlik uzmanları, bu saldırılara karşı korunma yöntemleri geliştirmek için sürekli çalışmaktadır. Bu yöntemler arasında trafiği izleme, anormallikleri tespit etme, saldırı trafiğini filtreleme ve dağıtık savunma mekanizmaları bulunur.

Otomata

Otomatlar, matematik ve bilgisayar biliminin temel kavramlarından biri olarak, belirli kurallar setine göre girdileri alıp, belirlenen durumlar arasında geçiş yaparak çıktılar üreten soyut makinelerdir. Başka bir deyişle, otomatlar, bir sistem veya sürecin belirli bir zamanda alabileceği durumları ve bu durumlar arasındaki geçişleri modellemek için kullanılır. Bu özellikleriyle otomatlar, ağ güvenliği gibi karmaşık sistemlerin analizi ve tasarımında değerli bir araç haline gelmiştir.

Ağ güvenliği bağlamında, otomatlar, ağ trafiğinin ve sistem davranışlarının modellemesi için kullanılır. Bu modelleme, normal ve anormal davranışlar arasındaki farkları tanımlamaya yardımcı olur, böylece güvenlik ihlalleri veya saldırı girişimleri, özellikle de Hizmet Dışı Bırakma (DoS) ve Dağıtık Hizmet Dışı Bırakma (DDoS) saldırıları gibi zararlı aktiviteler daha etkin bir şekilde tespit edilebilir. Otomatlar, ağ trafiği desenlerinin ve davranışlarının belirlenmesinde kullanılarak, güvenlik sistemlerinin, bu desenlerdeki sapmaları algılayıp uygun güvenlik önlemlerini devreye sokmasına olanak tanır.

Sonlu durum makineleri (FSM), ağ güvenliğinde sıkça kullanılan otomat türlerinden biridir. FSM'ler, ağ trafiğini ve sistem etkileşimlerini temsil eden durumlar ve bu durumlar arasındaki geçişlerle tanımlanır. Bu yapı, ağ güvenlik sistemlerinin, önceden tanımlanmış saldırı imzalarını veya anormal trafik desenlerini tanımasını ve buna göre alarm vermesini sağlar.

OTOMATLAR İLE DOS SALDIRI TESPİTİ

Amaç ve Ders ile İlişkisi

Gerçek zamanlı veri akışları, özellikle web hizmetleri ve bulut tabanlı uygulamalar açısından, günümüz dijital dünyasının temel taşlarından biridir. Bu akışlar, kullanıcı etkileşimlerinden büyük veri analizlerine kadar her şeyi mümkün kılar. Ancak, bu veri yollarının açıklığı, onları çeşitli siber saldırı türlerine, özellikle de Hizmet Dışı Bırakma (DoS) ve Dağıtık Hizmet Dışı Bırakma (DDoS) saldırılarına karşı savunmasız kılar. Bu saldırılar, hedeflenen sistemleri meşru kullanıcı trafiğinden mahrum bırakarak işlevsiz hale getirir. Ancak, Sonlu Durum Makineleri (FSM) kullanarak bu tür saldırıları tespit etmek, ağ güvenliği alanında umut verici bir yaklaşım sunmaktadır.

FSM, belirli bir sayıda durumdan oluşan bir modeldir ve her bir durum belirli koşullar altında bir sonraki duruma geçişi tanımlar. Bu özellik, FSM'yi gerçek zamanlı veri akışlarını analiz etmek ve anormal davranışları, özellikle de DoS saldırılarını tanımlamak için ideal kılar. Örneğin, bir web sunucusu normal koşullar altında farklı istek türlerine ve hacimlerine göre belirli durumlar arasında geçiş yapar. FSM modeli, bu normal geçiş desenlerini öğrenebilir ve beklenmeyen, anormal trafiği belirleyebilir, bu da potansiyel bir DoS saldırısının işareti olabilir.

FSM tabanlı sistemler, trafiğin davranışını sürekli olarak izleyerek ve analiz ederek gerçek zamanlı veri akışlarından gelen DoS saldırılarını tespit edebilir. Bu sistemler, belirli bir eşik değerinin üstünde bir trafik hacmi veya beklenmedik trafik desenleri gibi anormallikleri belirleyerek alarm verebilir. Bu yaklaşımın gücü, yüksek özelleştirilebilirliğinde ve adaptasyon kabiliyetindedir. Herhangi bir ağ ortamına özel olarak uyarlanabilir ve zamanla değişen trafik desenlerine adapte olabilir.

FSM kullanarak DoS saldırılarını tespit etmek, ağ güvenliğini artırmanın yanı sıra, sistemlerin sürekli olarak gelişen tehditlere karşı dayanıklılığını da sağlar. Bu yöntem, ağ yöneticilerine ve güvenlik uzmanlarına, saldırıları gerçekleşmeden önce tespit etme ve müdahale etme olanağı tanıyarak, kritik hizmetlerin sürekli olarak kullanılabilir kalmasını sağlar. Sonuç olarak, FSM tabanlı DoS saldırı tespit sistemleri, dijital altyapıları korumak için önemli bir araç olarak ön plana çıkmaktadır.

Yöntemler

Ağ trafiği desenlerini analiz ederek DoS saldırılarına özgü davranış kalıplarını tanımlamak ve bu kalıpları tanımlamak için düzenli ifadeler ve sonlu otomatlar geliştirip, gerçek zamanlı ağ trafiği verisi üzerinde bu otomatları test etmek, ağ güvenliğinin önemli bir yönünü oluşturur. Bu süreci etkili bir şekilde gerçekleştirebilmek için izlenebilecek yöntemler şunlardır:

-Gerçek zamanlı ağ trafiği verisini yakalamak için ağ analiz araçlarından (Wireshark, tcpdump vb.) yararlanılabilir.

-Normal ağ trafiği desenleri ile DoS saldırılarına özgü anormal davranışları ayırt etmek için trafik özellikleri (paket boyutu, istek sıklığı ve hedef adresleri gibi) incelenebilir.

-Yüksek trafiğin aniden artışı, tekrar eden istekler veya beklenmedik kaynaklardan gelen trafiğin artışı gibi DoS saldırı özelliklerini tanımlayın.

-Tanımlanan davranış kalıplarını temsil etmek için düzenli ifadeler geliştirilebilir.

-Tanımlanan düzenli ifadeleri temel alarak, DoS saldırı kalıplarını tanıyabilecek sonlu durum makineleri (FSM) tasarlanabilir.

-Tasarlanan sonlu durum makineleri test edilir ve gerekli güncellemeler yapılabilir.

Proje Kapsamında Yapılacaklar

Açıklanan yöntemler doğrultusunda DoS saldırıları sonucu oluşan ağ anomalilerinin tespitini yapacak otomat geliştirmek veya projenin seyrine bakılarak bu yöntemleri kullanan halihazırda bulunan otomatların detaylı bir şekilde açıklanması amaçlanmaktadır. DoS saldırı alt türlerinden olan SYN-Flood atak türünün tespiti amaçlanmaktadır. Bunun için Kali Linux üzerinden 'hping3' aracılığıyla basit bir atak senaryosu gerçekleştirilmesi planlanmıştır.

Görsel 1. SYN-Flood Gerçekleştiren Atak Kodu

Yukarıdaki görüntüde Kali Linux işletim sistemi üzerinden SYN atağı gerçekleştirmeye hazır hping3 kodu yer almaktadır. Bu kod ile beraber **15000 paket gönderimi (-c 15000), her biri 120 baytlık(-d 120), SYN bayrağı taşıyan (-S), TCP pencere genişliği 64 olan (-w 64), port 80'e gönderilen (-p 80), olabildiğinde en hızlı şekilde gönderilen(--flood) ve rastgele IP'ler üzerinden gönderimi sağlanan (--rand-source)** paketler oluşturulmaktadır.

Ardından Python'da ağ analizi yapmaya olanak sağlayan Scapy kütüphanesi kullanılarak atağın tespiti yapılacaktır. Gelen paketleri filtreleyip SYN bayrağı taşıyan TCP protokolü kullanan paketlere ait tokenler oluşturulduktan sonra yine Python üzerinde oluşturacak olan TDFA ile atağın gerçekleşip gerçekleşmeyeceği belirlenecektir.

Çözüme Yönelik Farklı Yaklaşımlar

Eşik değer tabanlı tespit ile normal ağ trafiği desenlerinin belirlenmesi ve belirli bir eşik değerin üzerine çıkılması durumunda bir saldırının tespit edilmesi sonucunda eğer ağ trafiği beklenenden fazla artarsa, otomatik olarak bir uyarı veya alarm oluşturulabilir.

Protokol analizi ile ağ protokollerinin kullanımını izlemek ve analiz etmek, DOS saldırılarını tespit etmek için bir yol sağlar. Örneğin, aynı kaynaktan gelen aşırı miktarda istekleri algılayarak bu isteklerin bir saldırı olup olmadığını belirlemek mümkündür.

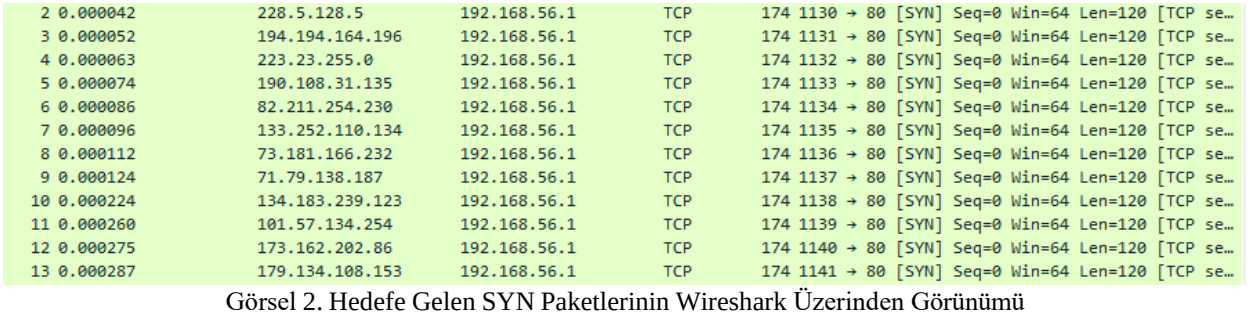
Paket incelemesi ve içerik filtreleme, ağ trafiğinin paket seviyesinde incelenmesi ve zararlı içeriğin veya saldırı işaretlerinin filtrelenmesi, saldırıları tespit etmek için başka bir yöntemdir. Örneğin, belirli imza tabanlı saldırı türlerini tespit etmek için paket içeriği imzaları kullanılabilir.

Zaman Serisi Analizi ise ağ trafiğinin zaman serileri verileri üzerinde analiz yapmak, normal ve anormal davranışların belirlenmesine yardımcı olur. Zaman serisi analizi, belirli zaman aralıklarında trafiğin özelliklerini izlemek ve anormal değişiklikleri tespit etmek için kullanılabilir.

PROBLEMİN ÇÖZÜMÜ

Gelen Paketlerin Tespiti

Gelen paketleri yukarıda da belirtildiği üzere Scapy yardımıyla tespiti yapılacaktır. Scapy'nin yanı sıra Wireshark aracılığıyla gelen paketleri daha detaylı bir şekilde görüntüleyebiliriz.



2	0.000042	228.5.128.5	192.168.56.1	TCP	174 1130 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
3	0.000052	194.194.164.196	192.168.56.1	TCP	174 1131 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
4	0.000063	223.23.255.0	192.168.56.1	TCP	174 1132 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
5	0.000074	190.108.31.135	192.168.56.1	TCP	174 1133 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
6	0.000086	82.211.254.230	192.168.56.1	TCP	174 1134 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
7	0.000096	133.252.110.134	192.168.56.1	TCP	174 1135 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
8	0.000112	73.181.166.232	192.168.56.1	TCP	174 1136 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
9	0.000124	71.79.138.187	192.168.56.1	TCP	174 1137 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
10	0.000224	134.183.239.123	192.168.56.1	TCP	174 1138 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
11	0.000260	101.57.134.254	192.168.56.1	TCP	174 1139 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
12	0.000275	173.162.202.86	192.168.56.1	TCP	174 1140 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…
13	0.000287	179.134.108.153	192.168.56.1	TCP	174 1141 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP se…

Görsel 2. Hedefe Gelen SYN Paketlerinin Wireshark Üzerinden Görünümü

Yukarıda görüldüğü üzere atak yapılan hedefte Wireshark çalıştırıldığı zaman ağ trafiğine bakılıp gerekli filtrelendimeler yapıldığı zaman gelen SYN paketleri görüntülenmektedir. Veriler incelendiğinde rastgele IP ler üzerinden hedefe SYN paketleri gönderimi sağlanıyor ve dikkatli bakıldığında bu paketler çok sık aralıklarla gönderiliyor ve atağın gerçekleşmesini sağlıyor.

Proje kapsamında paketlerin tespiti Wireshark yerine Spacy üzerinden yapılacaktır. Bunun nedeni ise gelen paketlerin Python üzerinden daha verimli bir şekilde analiz edilip daha kolay incelenmesidir.

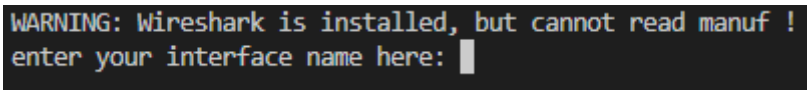
Atak Tespit Uygulaması

```
syn_tokens = []
start_time = None

def process_packet(packet):
    global start_time
    if start_time is None:
        start_time = packet.time

    if packet.haslayer(TCP) and packet[TCP].flags == "S":
        src_ip = packet[IP].src
        dst_ip = packet[IP].dst
        pck_flag = packet[TCP].flags
        timestamp = packet.time - start_time
        syn_token = {"src_ip": src_ip, "dst_ip": dst_ip, "timestamp": timestamp, "pck_flag": pck_flag}
        syn_tokens.append(syn_token)
        if len(syn_tokens) > 10:
            dfa.transition('a')
            if syn_tokens[-1]["timestamp"] - syn_tokens[-2]["timestamp"] < 0.1:
                dfa.transition('c')
                if dfa.is_accepting():
                    for i, token in enumerate(syn_tokens, 1):
                        print(f"{i}. {token}")
                    print("DDoS attack detected!!!")
                    exit(0)
            else:
                dfa.transition('b')
        else:
            dfa.transition('b')

input_interface = input("enter your interface name here: ")
sniff(iface=input_interface, prn=process_packet)
```

Görsel 3. Problemin Çözümü İçin Kullanılacak Kod-1

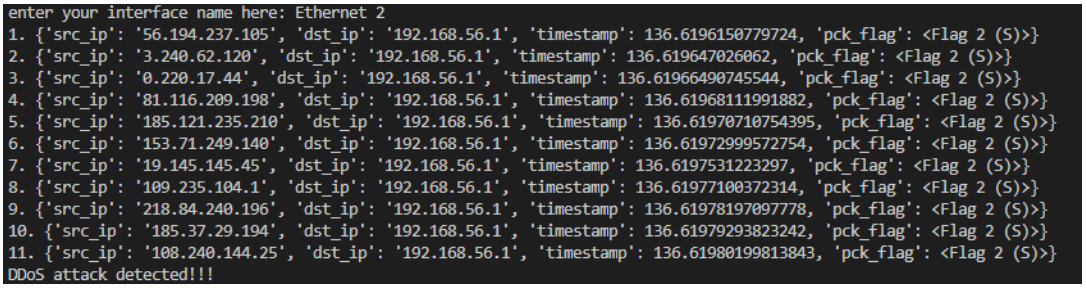
Yukarıdaki görselde Scapy kullanılarak paket analizini yapan Python kodu yer almaktadır. Öncelikle kullanıcıdan hangi ağ üzerinde inceleme yapılmasını belirlemek için ağın ismi istenmektedir. Bu ağ adı bilgisi ise Windows bilgisayarlarda CMD üzerinden “ipconfig”, Linux tabanlı sistemlerde ise “ifconfig” komutları aracılığıyla elde edilebilir.

‘sniff()’ fonksiyonu ile anlık ağ paketleri incelenip ‘process_packet’ sınıfına ağ paketleri gönderilmektedir. ‘process_packet’ fonksiyonu paketleri tek tek inceleyerek TCP protokolünü kullanan SYN bayrağı taşıyan paketleri filtreleyip onlara ait tokenler oluşturmaktadır. Sonrasında TDFA ya uyulup SYN paketleri incelenerek atağın gerçekleşip gerçekleşmediği belirlenir.



Görsel 4. Uygulama başladığında konsol ekranı

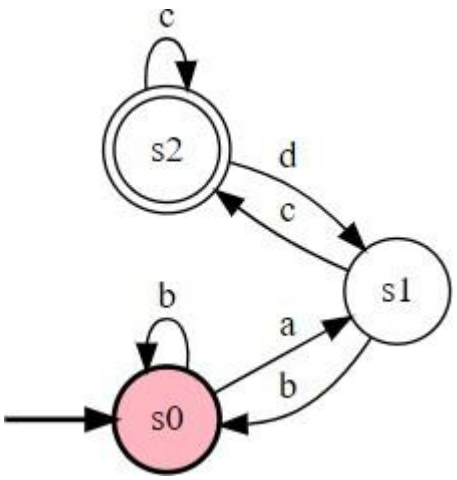
Bu kısımda kodu çalıştırdığımız zaman uygulama incelemek istediğimiz ağın ismini istiyor. Birden çok ağ olma ihtimaline karşın bu önlem alınmıştır. Windows kullanıcılarda ağ isimleri "Ethernet", "Ethernet 2", Linux ta ise "eth-0", "wlp3s0" gibi isimlerden biridir. Bunu Windows kullanıcıları, CMD üzerinden "ipconfig" komutu, Linux kullanıcıları ise konsol üzerinden "ifconfig" üzerinden öğrenebilir.



Görsel 5. Atak Tespit Edildiğinde Uygulamanın Konsol Ekranı

Uygulama, atak tespit etmesi durumunda atağa ait ilk 10 paket, konsol ekranına detaylarıyla yansıtılır ve atak olduğunu belirten bir mesaj gösterilir. Pakette yer alan bilgiler sırasıyla; "kaynak IP, hedef IP, kod çalışmasından itibaren geçen süre, pakete ait bayrak tipi" şeklindedir.

Geliştirilen TDFA Modeli



Görsel 6. Atak Tespiti İçin Geliştirilen TDFA

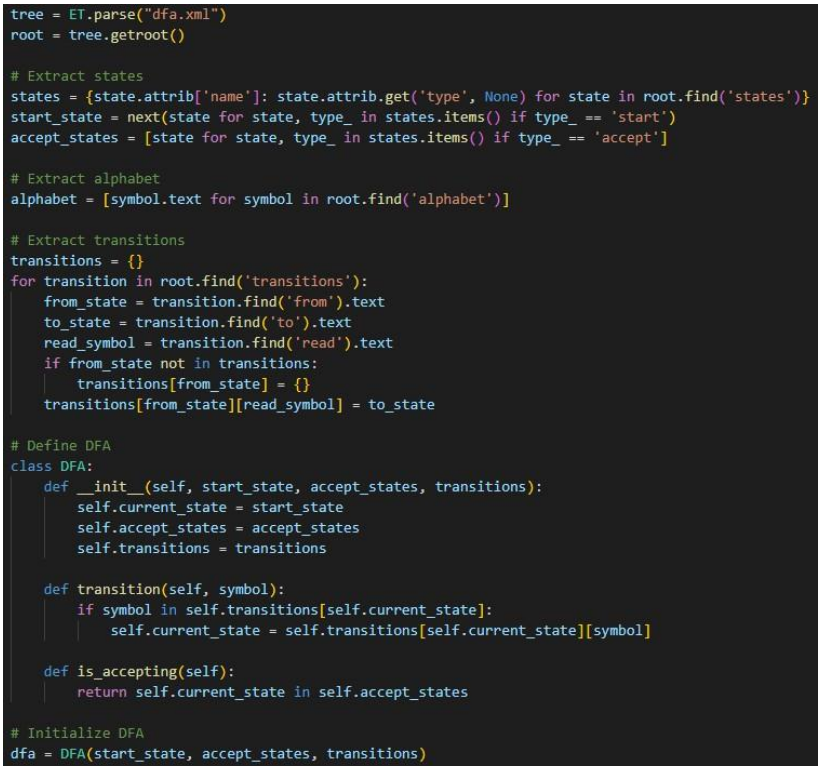
Görsel 4'te yer alan TDFA kullanılarak atağın tespiti hedeflenmektedir. Bu otomata ise Görsel 5'de Python ile yazılmıştır.
Bu otomata şunlardan oluşur;
Durumlar:
S0 => Başlangıç Durumu
S1 => Kritik Durum
S2 => DoS Atak Durumu
Alfabe:
a => SYN Paket Adeti > 10
b => SYN Paket Adeti <= 10
c => İki paket arası zaman aralığı < 0.1sn
d => İki paket arası zaman aralığı >= 0.1sn

Başlangıçta Görsel 3'te gelen paketlere ait tokenler oluşturulup 'syn_tokens' dizisine eklenir. Sonrasında
Bu dizide yer alan paketlerin sayısı kontrol edilir. Eğer gelen paket sayısı 10'dan fazlaysa Kritik Durumuna geçiş yapılır. Yoksa başlangıç durumu devam eder. Sonrasında eğer Kritik Duruma geçiş yapıldıysa tokenlerde yer alan paketlerin teslim edilme zamanı bilgisi kullanılarak iki paket arası süre kontrol edilir. Eğer bu süre 0.1 saniyeden az ise DoS Atak Durumuna geçilir ve kullanıcıya atak gerçekleştiğini söyleyen bir mesaj belirtilir. Yoksa Kritik Durum devam eder.

```xml
<dfa>
    <states>
        <state name="s0" type="start"/>
        <state name="s1"/>
        <state name="s2" type="accept"/>
    </states>
    <alphabet>
        <symbol>a</symbol>
        <symbol>b</symbol>
        <symbol>c</symbol>
        <symbol>d</symbol>
    </alphabet>
    <transitions>
        <transition>
            <from>s0</from>
            <to>s0</to>
            <read>b</read>
        </transition>
        <transition>
            <from>s0</from>
            <to>s1</to>
            <read>a</read>
        </transition>
        <transition>
            <from>s0</from>
            <to>s1</to>
            <read>b</read>
        </transition>
        <transition>
            <from>s1</from>
            <to>s2</to>
            <read>c</read>
        </transition>
        <transition>
            <from>s1</from>
            <to>s2</to>
            <read>d</read>
        </transition>
        <transition>
            <from>s2</from>
            <to>s2</to>
            <read>c</read>
        </transition>
    </transitions>
</dfa>
```

Görsel 7. Atak Tespiti İçin Geliştirilen DFA'nın XML İle Yazılmış Hali

Yukarıdaki görselde anlatılan otomatanın XML dosyasına yazılmış hali yer almaktadır. Bu XML'de başlangıçta state'ler belirlenmiş olup, sonrasında alfabe ve son olarak "transitions" başlığı altında tüm geçişler tanımlanmımştır.



```python
tree = ET.parse("dfa.xml")
root = tree.getroot()

# Extract states
states = {state.attrib['name']: state.attrib.get('type', None) for state in root.find('states')}
start_state = next(state for state, type_ in states.items() if type_ == 'start')
accept_states = [state for state, type_ in states.items() if type_ == 'accept']

# Extract alphabet
alphabet = [symbol.text for symbol in root.find('alphabet')]

# Extract transitions
transitions = {}
for transition in root.find('transitions'):
    from_state = transition.find('from').text
    to_state = transition.find('to').text
    read_symbol = transition.find('read').text
    if from_state not in transitions:
        transitions[from_state] = {}
    transitions[from_state][read_symbol] = to_state

# Define DFA
class DFA:
    def __init__(self, start_state, accept_states, transitions):
        self.current_state = start_state
        self.accept_states = accept_states
        self.transitions = transitions

    def transition(self, symbol):
        if symbol in self.transitions[self.current_state]:
            self.current_state = self.transitions[self.current_state][symbol]

    def is_accepting(self):
        return self.current_state in self.accept_states

# Initialize DFA
dfa = DFA(start_state, accept_states, transitions)
```

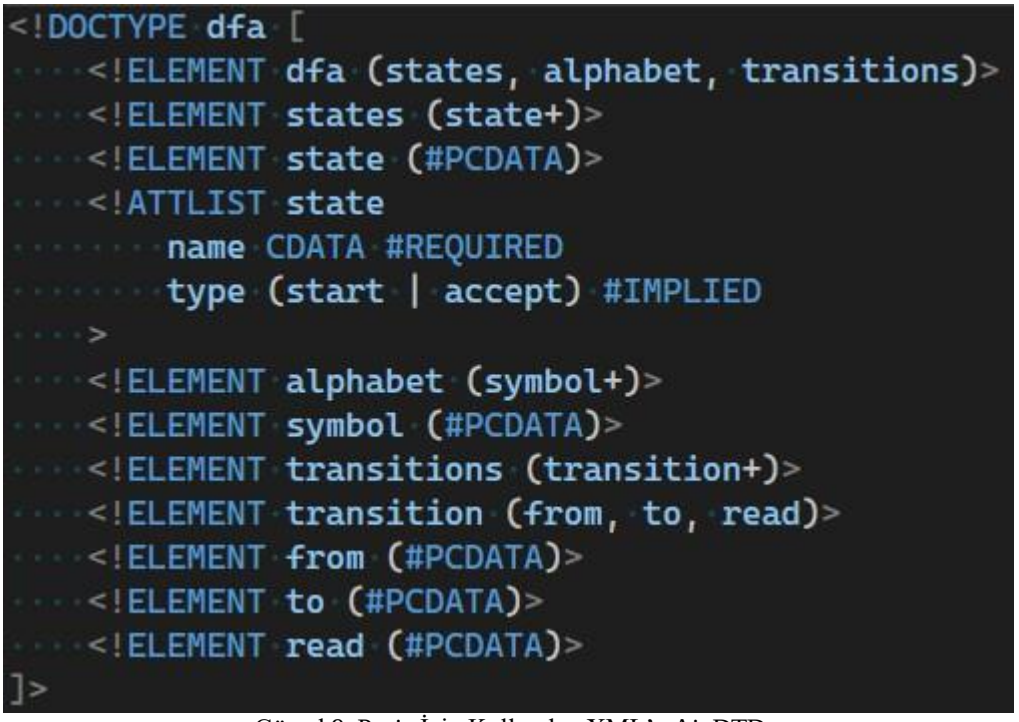
Görsel 8. XML Dosyasının Koda Aktarılmasını Sağlayan Kod

Yukarıdaki Python kodu, XML formatında tanımlanmış DFA'yı okuyarak bir DFA nesnesi oluşturur ve işlevlerini gerçekleştirir. Kodun ilk adımı, `dfa.xml` dosyasını XML olarak okur ve kök elemanına (`root`) erişir. Ardından, XML dosyasındaki tüm durumları (`states`) ve bu durumların tiplerini (başlangıç veya kabul durumu) çıkarır. Bu işlem, `states` sözlüğü oluşturularak yapılır; burada anahtarlar durum adlarını, değerler ise durumların tiplerini belirtir. Başlangıç durumu (`start_state`) ve kabul durumları (`accept_states`) bu sözlükten belirlenir.

Kod, DFA'nın alfabesini de çıkarır. Alfabe, XML dosyasındaki `alphabet` elemanından alınan sembollerin bir listesidir. Sonrasında, DFA'nın geçiş kuralları (`transitions`) çıkarılır. Her geçiş, başlangıç durumu (`from_state`), hedef durumu (`to_state`) ve okunan sembol (`read_symbol`) bilgilerini içerir. Bu geçişler, `transitions` sözlüğünde başlangıç durumuna göre gruplanmış ve sembollere göre hedef durumlara atanmıştır.

Bu bilgiler toplandıktan sonra, DFA'yı temsil eden `DFA` sınıfı tanımlanır. Bu sınıfın `__init__` metodu, DFA'yı başlangıç durumu, kabul durumları ve geçişlerle başlatır. `transition` metodu, belirtilen sembole göre geçiş yapar ve `is_accepting` metodu, DFA'nın şu anda kabul durumunda olup olmadığını kontrol eder.

Son olarak, yukarıda çıkarılan başlangıç durumu, kabul durumları ve geçişlerle bir `DFA` nesnesi (`dfa`) oluşturulur. Bu nesne, sembollere göre geçiş yapabilir ve belirli bir giriş dizisinin DFA tarafından kabul edilip edilmediğini belirleyebilir. Bu kod, DFA'nın çalışmasını simüle etmek ve otomatonun belirli bir girdiyi kabul edip etmediğini test etmek için kullanılabilir. Böylece, DFA'nın belirli bir giriş dizisini kabul edip etmediği, Python kodu aracılığıyla kolayca kontrol edilebilir.



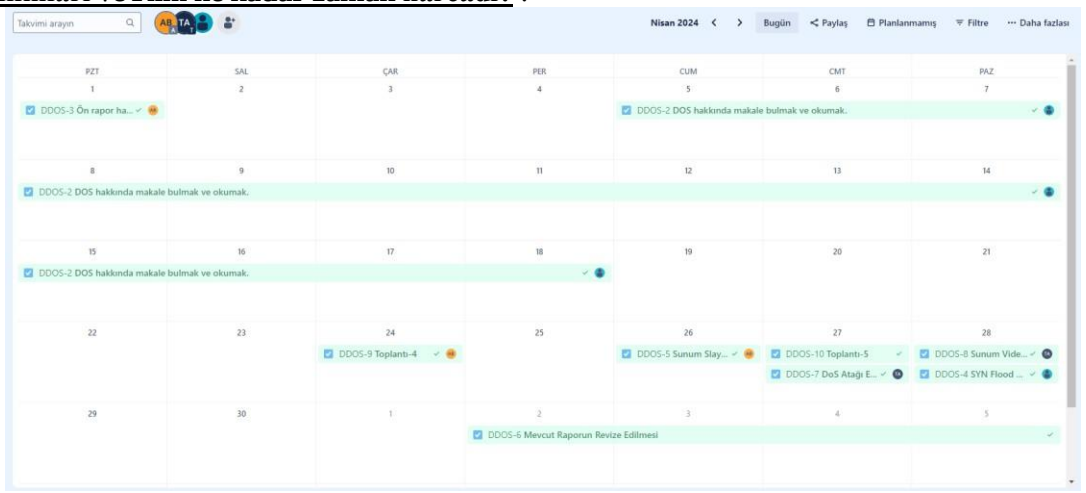
Görsel 9. Proje İçin Kullanılan XML'e Ait DTD

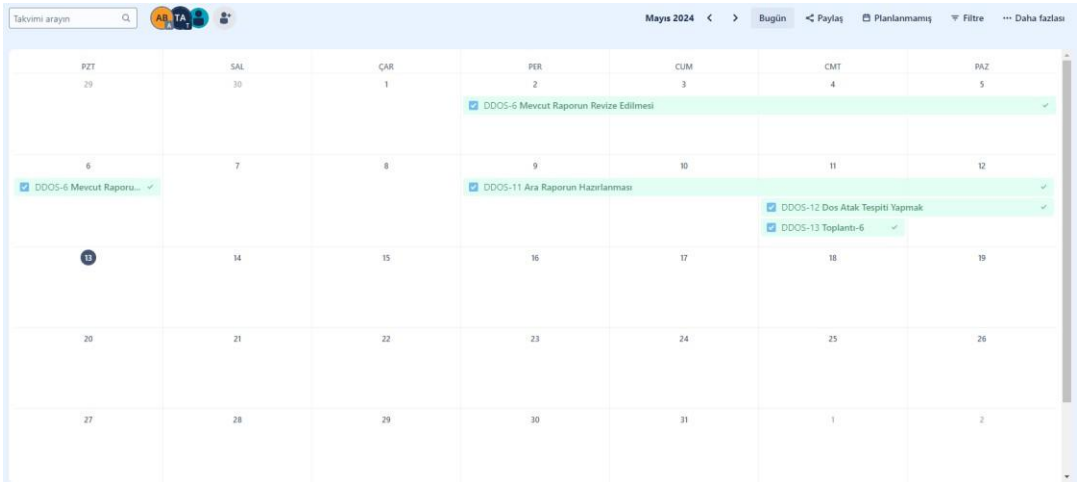
Yukarıdaki DTD(Document Type Definition), proje kapsamında kullanılacak DFA'nın XML formatında nasıl tanımlanacağını belirler. DFA yapısı üç ana bileşenden oluşur: durumlar (states), alfabe (alphabet) ve geçişler (transitions). Her durum, bir isim (name) ve opsiyonel olarak bir tip (type) özniteliğine sahiptir. Alfabe, bir veya daha fazla sembol içerir. Geçişler, her biri başlangıç durumu (from), bitiş durumu (to) ve okunan sembolü (read) içeren bir dizi geçişten oluşur. Bu yapı, DFA'nın durumlarını, girdilerini ve geçiş kurallarını net bir şekilde tanımlamayı sağlar.

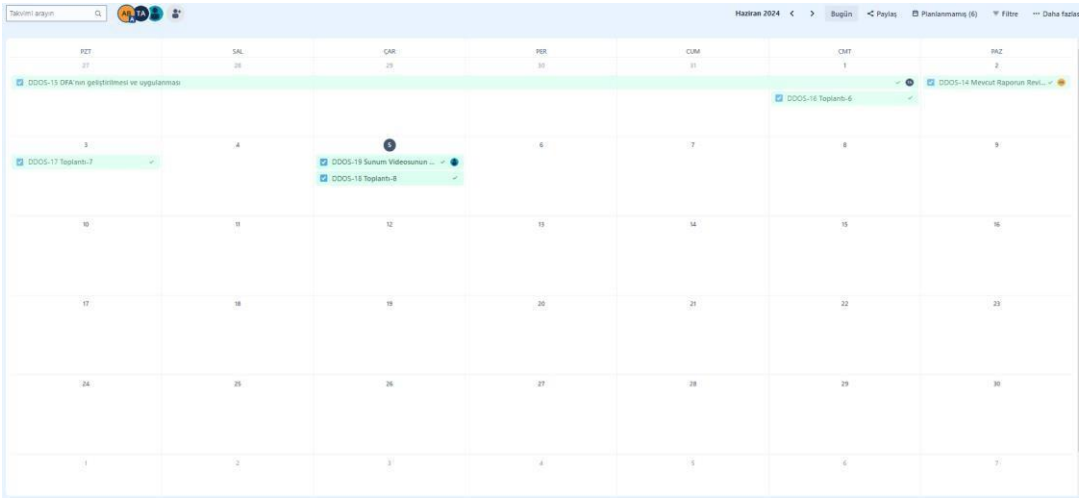
PROJE EKİBİ DEĞERLENDİRMESİ

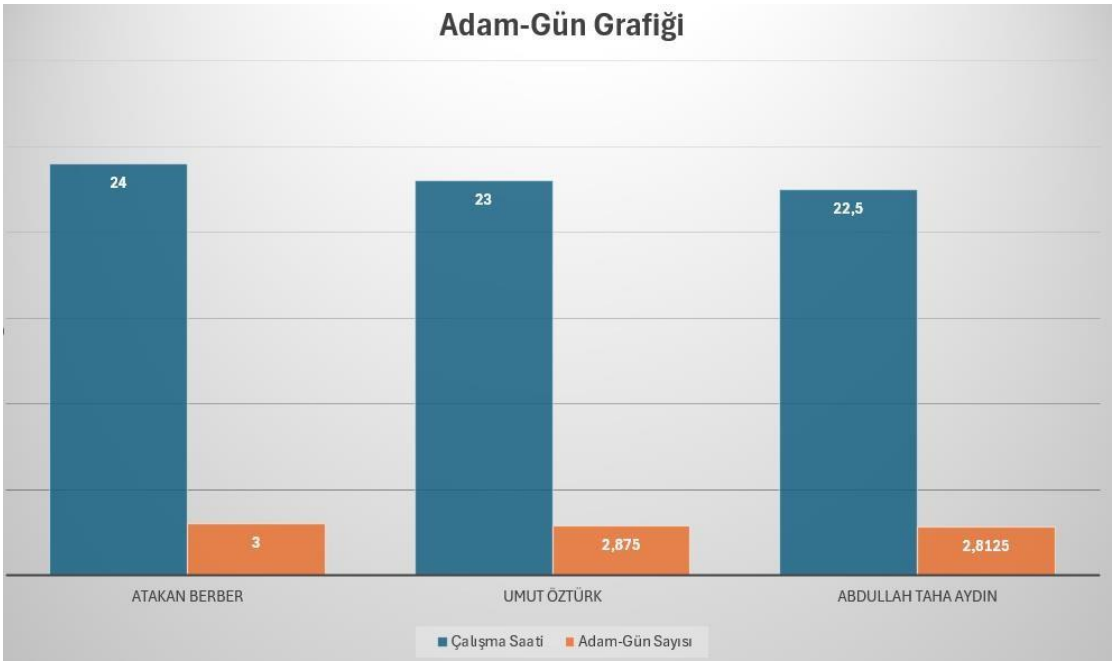
Grup koordinatörü : 152120211052 – Umut Öztürk

İş dağılımları ve Kim ne kadar zaman harcadı? :









Toplantı Detayları:

Nisan ayı içerisinde 24 Nisan ve 27 Nisan tarihlerinde Discord üzerinden 4'er saat iki(2) toplantı yapılmıştır.

Mayıs ayı içerisinde 12 Mayıs tarihinde Discord üzerinden 4 saat süren bir(1) toplantı yapılmıştır.

Haziran ayı içerisinde 1 Haziran, 3 Haziran ve 5 Haziran tarihlerinde 3 saat süren üç(3) toplantı yapılmıştır.

İş Dağılımı:

Abdullah Taha Aydın: Mevcut DFA'nın geliştirilmesi ve düzenlenmesi

Umut Öztürk : XML, DTD hakkında bilgi edinme ve rapor/video düzenlenmesi

Atakan Berber : DFA'nın XML e implemente edilmesi ve kod tarafından okunması

KAYNAKÇA

- https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/06/tcpdump-kullan%C4%B1m%C4%B1

- "Introduction to Automata Theory, Languages, and Computation" by Hopcroft, Motwani and Ullman

- Stallings, W. (2017). Cryptography and Network Security: Principles and Practice.

- https://www.investopedia.com/terms/d/denial-service-attack-dos.asp

- Zargar, S. T., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. IEEE Communications Surveys & Tutorials

- Mirkovic, J., & Reiher, P. (2004). A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. ACM SIGCOMM Computer Communication Review

- The Study on Network Attacks based on Automaton Theory by SHANG Qing-wei, CAO Ke and WANG Feng

- Runtime Verification for Anomaly Detection of Robotic Systems Security by Yunus Sabri Kirca ,Elif Degirmenci , Zekeriyya Demirci ,Ahmet Yazici, Metin Ozkan, Salih Ergun and Alper Kanak

- SYN Flood DoS Detection System Using Time Dependent Finite Automata by Noura AlDossary, Sarah AlQahtani, Reem Alzaher and Atta-ur-Rahman

- https://www.firewall.cx/tools-tips-reviews/network-protocol-analyzers/performing-tcp-syn-flood-attack-and-detecting-it-with-wireshark.html

- https://gelecegiyazanlar.turkcell.com.tr/blog/tcp-syn-ve-tcp-synack-ddos-saldirilari