TCP SYN SELİ SALDIRISININ ETKİLERİNİ AZALTMAK İÇİN YENİ SYN ÇEREZLERİ GERÇEKLEMESİ

Bu tez Bilgi Güvenliği Mühendisliği'nde
Tezli Yüksek Lisans Programının bir koşulu olarak

İbrahim ÇELİKBİLEK

tarafından

Fen Bilimleri Enstitüsü'ne
sunulmuştur.

İSTANBUL
ŞEHİR
ÜNİVERSİTESİ

Bu tezi okuduk. kapsam ve nitelik açısından Bilgi Güvenliği Mühendisliği alanında Yüksek Lisans derecesi için tümüyle uygun olduğu görüşüne vardık.

ONAYLAYANLAR:

Dr. Mehmet Kara
(Tez Danışmanı) .....

Prof. Dr. Tahsin Erkan Türe .....

Doç. Dr. Ali Gökhan Yavuz .....

Bu tez İstanbul Şehir Üniversitesi, Fen Bilimleri Enstitüsü tarafından belirlenen tüm koşullara uygundur.

ONAY TARİHİ:

MÜHÜR/İMZA:

Yazarlık Beyanı

Ben, İbrahim ÇELİKBİLEK, başlığı, 'TCP SYN SELİ SALDIRISININ ETKİLERİNİ AZALTMAK İÇİN YENİ SYN ÇEREZLERİ GERÇEKLEMESİ' olan tezin ve içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlilik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Esaslı yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

İmza: 

Tarih: 10.02.2016

TCP SYN SELİ SALDIRISININ ETKİLERİNİ AZALTMAK İÇİN YENİ SYN ÇEREZLERİ GERÇEKLEMESİ

İbrahim ÇELİKBİLEK

Özet

Günümüzde ticari ya da kamusal web siteleri ve bunların çevrim içi hizmetleri Servis Dışı Bırakma (DoS-Denial of Service) veya Dağıtık Servis Dışı Bırakma (DDoS-Distributed Denial of Service) saldırılarından kaynaklanan risklerle karşı karşıya bulunmaktadırlar. DoS/DDoS en basit ve etkili siber saldırı yöntemlerinden biri haline gelmiştir.

Tezimiz bir DDoS türü olan TCP (Transmission Control Protocol) SYN seli saldırısına odaklanmıştır. TCP sunucularını hedef alan bu saldırı türünün amacı sunucuların erişebilirliğini ortadan kaldırmaktır. Tez kapsamında ilk olarak bu saldırı türünden korunmak ve etkilerini azaltmak için literatürde bulunan yöntemler anlatılmıştır.

D.J.Bernstein tarafından TCP SYN seli saldırısının etkilerini azaltmak için önerilen SYN çerezleri yöntemi ve bu yöntemin Linux çekirdeği üzerindeki gerçeklemesi ayrıntılı bir şekilde incelenmiştir. Yöntemin güvenliğinin arttırılması ve TCP SYN seli saldırısının tespit edilmesi ile ilgili olarak literatürde bulunan çalışmalar ve öneriler ayrıntılı olarak sunulmuştur.

Tez kapsamında SYN çerezleri yönteminin güvenliğinin arttırılması ve saldırı tespit yönteminin değiştirilmesi ile ilgili öneriler geliştirilmiş ve gerçeklemeleri yapılmıştır. Oluşturulan yeni SYN çerezleri yöntemi ile TCP SYN seli saldırısına karşı güvenliğin arttırılması ve kaynak (CPU,RAM) tüketiminin azaltılması hedeflenmiştir. Sonuç olarak temel hedef sunucuların erişilebilirliğinin arttırılmasıdır.

Önerdiğimiz yöntem sadece sunucu işletim sistemlerinin çekirdek kodları üzerinde değişiklik içermektedir. Bunun yanında kullanıcı ve sistem uygulamalarında herhangi bir değişiklik yapılması gerekmemektedir

Anahtar Sözcükler: Transmission Control Protocol (TCP), Hizmet Dışı Bırakma, Dağıtık Hizmet Dışı Bırakma, TCP SYN Seli Saldırısı, SYN Çerezleri, Linux Çekirdeği

NEW SYN COOKIES IMPLEMENTATION TO DECREASE THE EFFECTS OF TCP SYN FLOOD ATTACK

İbrahim ÇELİKBİLEK

Abstract

Today, commercial or civil web sites and their online services encounter with risks arising out of DoS (Denial of Service) or DDoS (Distributed Denial of Service) intrusions. DoS/DDoS became one of the most simple and effective cyber intrusion methods.

Our thesis focuses on the intrusion of TCP (Transmission Control Protocol) SYN flood which is a kind of DDoS. The aim of this intrusion which targets TCP servers is to remove the accessibility of the servers. In our thesis, we firstly touch upon the methods which are used in order to protect against this type of intrusion and to mitigate its effects and which are available in the literature.

SYN cookies method which was suggested by D.J.Bernstein in order to mitigate the effects of TCP SYN flood intrusion and the occurrence of this method on Linux kernel were discussed in detail. Studies and suggestions which are available in the literature with regard to increasing the security of method and determination of TCP SYN flood intrusion were also presented in detail.

In our thesis, some suggestions were developed and justified with regard to increasing the security of SYN cookies method and changing the method of intrusion determination. With the help of newly developed SYN cookies method, it was aimed at increasing the security against TCP SYN flood intrusion and decreasing the consumption of the source (CPU,RAM). In conclusion; it is aimed at increasing the accessibility of basic target servers.

Our recommended method just includes changes on the kernel codes of the server operating system. There is no need to make any change in user and system applications.

Keywords: Transmission Control Protocol (TCP), Denial Of Service (DoS), Distributed Denial Of Service (DDoS), TCP SYN Flood Attack, SYN Cookies, Linux Kernel

Teşekkür

Tez çalışmam boyunca bana sürekli yol gösteren ve yardımlarını esirgemeyen saygıdeğer hocam Sayın Dr. Mehmet KARA Bey'e ve benden hiçbir zaman desteğini esirgemeyen eşim Sayın Perihan ÇELİKBİLEK Hanım'a ve aileme teşekkürlerimi sunarım.

İçindekiler

Yazarlık Beyanı **ii**

Özet **iii**

Abstract **iv**

Teşekkür **v**

Şekil Listesi **viii**

Kısaltmalar **ix**

1 Giriş **1**

2 Servis Dışı Bırakma Saldırıları **2**
2.1 Servis Dışı Bırakma (Denial of Service) 2
2.2 Dağıtık Servis Dışı Bırakma (Distributed Denial of Service) 3
2.3 Köle Bilgisayarlar ve DDOS 4
2.4 Dünya Genelinde Yapılmış DDoS Saldırıları 4

3 DDoS Saldırı Çeşitleri **6**
3.1 UDP Seli 6
3.2 ICMP Seli 7
3.3 Parçalanmış Paket Atağı 8
3.4 Ping of Death 8
3.5 Slowloris 8
3.6 HTTP Get Seli 9
3.7 HTTP Post Seli 9
3.8 Slow Post 10
3.9 Apache Killer 10
3.10 DNS Seli ve DNS Sorgulama Saldırısı 10

4 DDoS Saldırılarından Korunmak İçin Alınması Gereken Önlemler **11**
4.1 Ağ Trafiğini İzlemek 12
4.2 Yük Dengeleyicileri Kullanmak 12
4.3 Saldırı Tespit ve Önleme Sistemleri Kurmak 12
4.4 Saldırıları Saptırmak 13
4.5 Yazılım Güncellemelerinin Yapılması 13
4.6 Bantgenişliğinin Arttırılması 13

4.7 Doğru Konfigürasyonun Yapılması 13
4.8 Yönlendirici Seviyesinde DDoS Saldırılarından Korunmak 14
4.9 Güvenlik Duvarı Seviyesinde DDoS Saldırılarından Korunmak 14

5 TCP SYN Seli Saldırısı ve Korunma Yöntemleri 15
5.1 TCP Protokolü İle İlgili Temel Bilgiler 15
5.2 TCP Oturum Başlatma ve Sonlandırma 19
5.3 TCP SYN Seli Saldırısı 23
5.4 TCP SYN Seli Saldırısından Korunmak İçin Kullanılan Mevcut Yöntemler 25
5.4.1 SYN Önbellek 26
5.4.2 SYN Vekil Sunucu/Güvenlik Duvarı 26
5.4.3 SYN Çerezleri 26
5.5 SYN Çerezleri Yönteminin İncelenmesi 29
5.6 SYN Çerezleri Yöntemini İyileştirmek İçin Önerilen Yöntemler 30
5.7 SYN Çerezleri Yöntemini İyileştirmek İçin Önerdiğimiz Yöntem 32
5.7.1 Güvenliğinin Arttırılması 32
5.7.2 Algılanma Yönteminin Değiştirilmesi 33
5.7.2.1 Mevcut 2.6.32.69 Çekirdek Gerçeklemesinin İncelenmesi . 33
5.7.2.2 Önerilen 2.6.32.69 Çekirdek Yapısının Gerçeklenmesi . . . 37

6 Test Ortamı ve Sonuçların Değerlendirilmesi 41
6.1 2.6.32.69 Çekirdek Sürümünün Derlenmesi ve Gerekli Programların Kurulması 42
6.2 Test İşleminin Yapılması ve Sonuçların Değerlendirilmesi 44
6.2.1 Sınırlandırılmış Paket Sayısı İle Saldırı 44
6.2.2 Sınırlandırılmamış Paket Sayısı İle Saldırı 46
6.2.3 Sonuçlar ve Öneriler 49

Şekil Listesi

2.1 DoS Atağı Senaryosu 2
2.2 DDoS Atağı Senaryosu 3
2.3 Köle Bilgisayarlar İle Yapılan DDoS Atağı Senaryosu 4

3.1 UDP Başlığı Formatı 7
3.2 ICMP Başlığı Formatı 7

5.1 TCP Protokol Başlığı 16
5.2 TCP Durum Diyagramı 18
5.3 Üç Yollu El Sıkışma 19
5.4 Üç Yollu El Sıkışma (Ayrıntılı) 20
5.5 TCP Bağlantısının Kurulması 21
5.6 TCP Oturum Sonlandırma 22
5.7 SYN Seli Saldırısı 23
5.8 SYN Seli Saldırısı Tespiti 25
5.9 SYN Çerezleri İle Üç Yollu El Sıkışma 27
5.10 İlk Sıra Numarası Bitsel Şablonu 27
5.11 Önerilen İlk Sıra Numarası Bitsel Şablonu 31
5.12 Önerdiğimiz İlk Sıra Numarası Bitsel Şablonu 32
5.13 Yeni Önerilen Sistemin Çalışma Modeli 38

6.1 Test Ortamında Kullanılan İşletim Sistemleri 41
6.2 Saldırı Topolojisi 44
6.3 Ortalama İşlemci Kullanım Oranları 45
6.4 Eski Sistem İşlemci Kullanım Oranları 45
6.5 Yeni Sistem İşlemci Kullanım Oranları 46
6.6 Saldırı Topolojisi 46
6.7 Ortalama İşlemci Kullanım Oranları 47
6.8 Eski Sistem İşlemci Kullanım Oranları 47
6.9 Yeni Sistem İşlemci Kullanım Oranları 48

Kısaltmalar

TCP	**T**ransmission **C**ontrol **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
IP	**I**nternet **P**rotocol
OSI	**O**pen **S**ystems **I**nterconnection
ISP	**I**nternet **S**ervice **P**rovider
ICMP	**I**nternet **C**ontrol **M**essaging **P**rotocol
DNS	Domain **N**ame **S**ystem
DoS	**D**enial **O**f **S**ervices
DDoS	**D**istrubuted **D**enial **O**f **S**ervices
SYN	**Syn**chronize
TCB	**T**ransmission **C**ontrol **B**lock
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
IRC	**I**nternet **R**elay **C**hat
TFTP	**T**rivial **F**ile **T**ransfer **P**rotocol
IDS	**I**ntrusion **D**etection **S**ystem
IPS	**I**ntrusion **P**reventation **S**ystem
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
MSS	**M**aximum **S**egment **S**ize

Bölüm 1

Giriş

DDoS saldırıları Web, DNS (Domain Name System), veri tabanı ve diğer sunucuların çevrim içi faaliyetlerini durdurmak için en etkili ve basit yöntemlerden biri olarak görülmektedir. Geleneksel güvenlik önlemleri DDoS saldırılarını önlemekte yetersiz kalmaktadır[1].

Günümüzde saldırganlar tarafından en çok tercih edilen DDoS saldırı türlerinden birisi TCP SYN seli olarak tespit edilmektedir[2]. TCP SYN seli TCP sunucusuna karşı yapılan servis dışı bırakma saldırısı olarak tanımlanabilir. Bu saldırı türünün maliyeti oldukça düşük olup kolayca yapılabilmektedir. Saldırı yapacak araçların kolayca elde edilmesi yanında istendiğinde bu saldırı araçları kiralanabilmektedir.

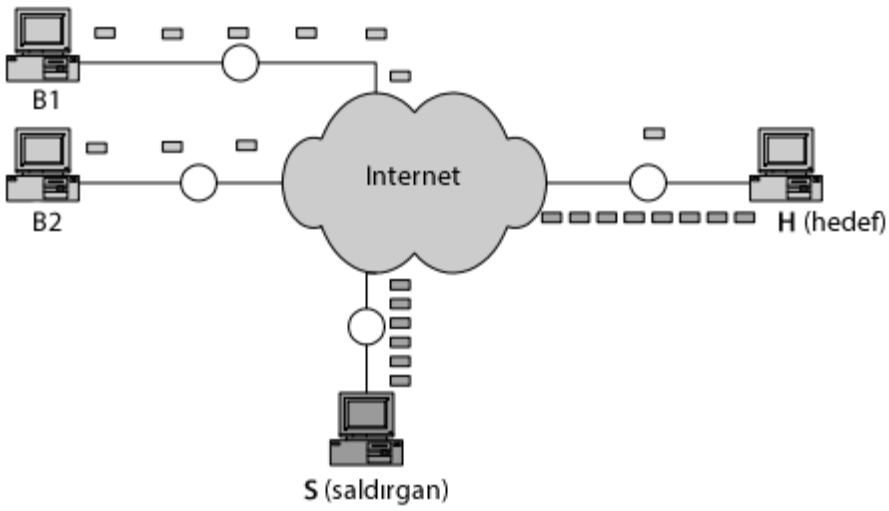
Tez kapsamında ilk önce DoS/DDoS saldırı türlerinin tamamı incelenmiş ve saldırıların etkilerini azaltmak için alınması gereken önlemler anlatılmıştır. Tezimiz SYN seli saldırısına odaklanmış ve bu saldırı türününün etkisini azaltmak için 1996 yılında D.J. Bernstein tarafından önerilen SYN çerezleri yöntemi ayrıntılı bir şekilde incelenmiştir[3]. Tezimizde bu yöntemin güvenliğinin arttırılması ve gerçeklenmesi ile ilgili yeni öneriler sunulmaktadır.

Bölüm 2

Servis Dışı Bırakma Saldırıları

2.1 Servis Dışı Bırakma (Denial of Service)

DoS sistemin ya da servislerin aşırı yüklenmesini sağlayarak hizmet veremez hale gelmelerini hedefleyen bir saldırı türüdür. Bu saldırı türünde hedef internet uygulamasının kendisi değil uygulamayı barındıran sunucu ve kaynaklardır. DoS saldırıları sonucunda sunucular ya da sistemler hizmet dışı bırakılabilir. Bunun sonucunda uygulama sahibi şirket ya da kişi; iş kaybı, maddi kayıp ve itibar kaybı yaşayabilir. Şekil 2.1'de görüldüğü üzere DoS saldırıları tek bir merkezden yine tek bir merkeze yapılan saldırılar olarak adlandırılır.

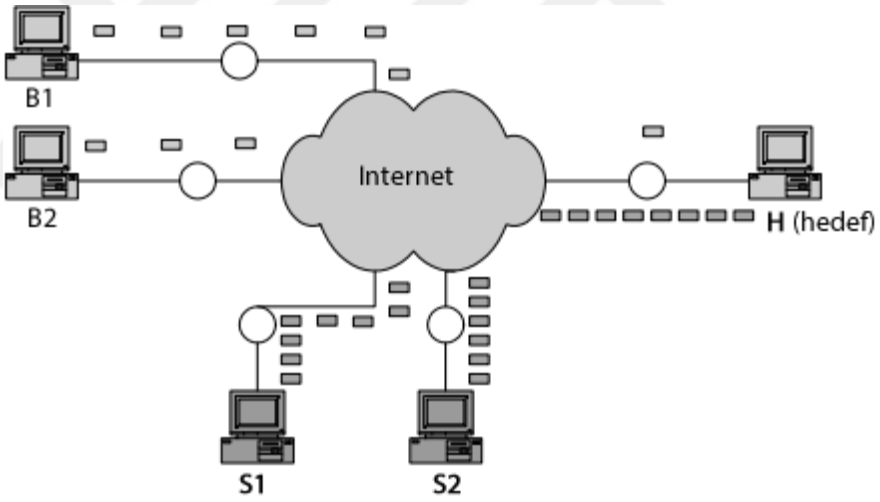


Şekil 2.1: DoS Atağı Senaryosu

2.2 Dağıtık Servis Dışı Bırakma (Distributed Denial of Service)

DDOS; İnternet üzerinden ele geçirilen birden fazla sistemden tek bir noktaya yapılan servis dışı bırakma saldırıları olarak tanımlanabilir. Bu tür saldırıların asıl amacı bilgi güvenliğinin en önemli yapı taşı olan erişebilirliği ortadan kaldırmaktır. DDoS saldırılarının DoS saldırılarından en önemli farkı birden fazla noktadan tek bir noktaya aynı anda yapılmalarıdır. Tek bir noktaya yapılan bu saldırılarda on binlerce bilgisayar taşeron olarak kullanılabilmektedir.

DDoS saldırı çeşidinde farklı noktalardan sunucuya eş zamanlı olarak çok sayıda veri paketi gönderilir. Sunucu kendisine gönderilen paketleri işlerken kendi sistem kaynaklarını (işlemci, hafıza, bantgenişliği) tüketebilir. DDoS saldırılarının temel amacı hedef sistemin bantgenişliğini ya da kaynaklarını (CPU, Ram, Disk) tüketmektir. Sonuç olarak sunucu veri paketlerine cevap veremez hale geldiğinde servis dışı kalmış olacaktır. Şekil 2.2'de DDoS saldırı senaryosu görülmektedir.



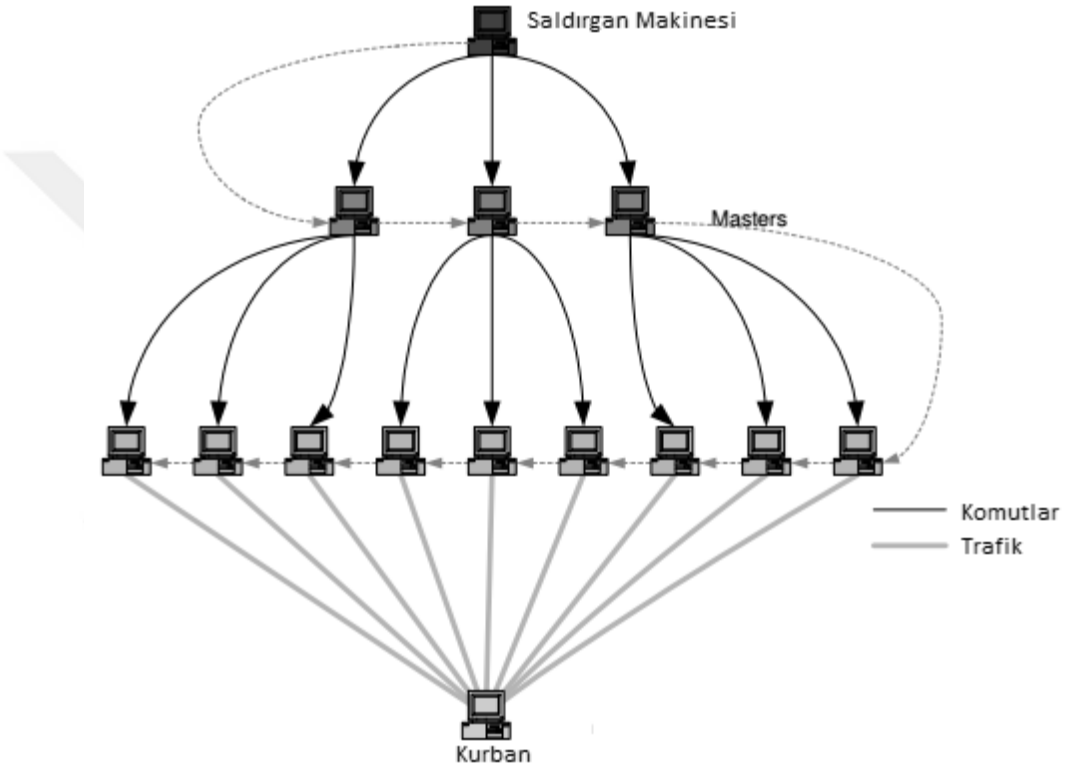
Şekil 2.2: DDoS Atağı Senaryosu

DDoS saldırılarına maruz kalan sistemin yanında ikincil kurban köleleştirilmiş bilgisayarlardır. Temel olarak tek bir IP adresinden yani sistemden gelen saldırılar donanımsal ya da yazılımsal olarak önlenebilir. Fakat birden fazla noktadan gelen saldırıları tespit etmek ve engellemek oldukça zordur. DDoS saldırılarında amaç sistemleri işlevsiz hale getirmek ve kullanımlarını engellemektir.

Saldırı tek bir IP adresinden gerçekleşirse bu IP adresi güvenlik duvarından tespit edilip engellenebilir. Fakat DDoS saldırılarında çok sayıda istemci kullanıldığından, IP tespiti güçleşmekte ve güvenlik duvarlarında log taşması oluşabilmektedir.

2.3 Köle Bilgisayarlar ve DDOS

DDoS saldırıları gerçekleştirmek için saldırganlar tarafından kullanılan bilgisayarlar köle (zombi) bilgisayar olarak adlandırılmaktadır. Köle bilgisayar toplulukları ise "BOTNET" olarak isimlendirilir. Saldırganlar bir çok köle bilgisayarı tek bir hedefe yönlendirebilirler. Botnetler; DDoS saldırıları yapmak, istenmeyen e-posta mesajları göndermek ve virüsleri yaymak gibi amaçlar için kullanılırlar. Şekil 2.3'de köle bilgisayar topluluğu ile yapılan bir DDoS atağı senaryosu görülmektedir.



Şekil 2.3: Köle Bilgisayarlar İle Yapılan DDoS Atağı Senaryosu

2.4 Dünya Genelinde Yapılmış DDoS Saldırıları

DoS saldırılarının geçmişi DDoS saldırılarına nazaran daha eskiye dayanmaktadır. Sadece tek bir merkezden yapılan saldırılar çok fazla etkili olmamaktadır. Bunun yanında DDoS saldırıları 1999 yılından itibaren gündeme gelmeye başlamıştır. İlk ciddi DDoS saldırısı 1999 Ağustos ayında meydana gelmiştir. Bu saldırıda Trino adı verilen bir DDoS saldırı aracı ile Minnesota Üniversitesine ait bir bilgisayar hedef olarak seçilmiştir.

Bu saldırıdan sonra 2000 yılında Amazon, Yahoo, eBay ve Datek şirketlerine DDoS saldırıları gerçekleştirilmiştir. Bu saldırılarda Yahoo, 3 saat boyunca son kullanıcıya

hizmet verememiştir. Şubat 2001 de İrlanda Ekonomi Bakanlığı sunucularına yapılan saldırıda sunucular büyük ölçüde zarar görmüşlerdir[4].

2002'de DNS Root Server'lara yapılan saldırı sonucunda 13 tane kök sunucudan dokuzu büyük ölçüde zarar görmüştür. 2007'de Wolfenstain, Counter Strike gibi oyunlarında aralarında bulunduğu on binden fazla oyun sunucusuna binden fazla bilgisayardan oluşan bir BOTNET ağıyla saldırı düzenlenmiş ve birçok bilgi ifşa olmuştur.

2008'de Gürcistan Cumhur Başkanlığı, Ulusal Bankası ve birçok devlet sitesi Rus korsanlarının DDoS saldırılarına maruz kalmıştır. 2010 yılında Mavimarmara, Wikileaks, Youtube olaylarını protesto etmek için yapılan saldırılar ve 2011 yılında PlayStation sunucularına yapılan DDoS saldırıları en dikkat çeken saldırılardır. 2012 yılında ABD Finans kuruluşlarına karşı çok ciddi DDoS atakları gerçekleştirilmiştir. Bu saldırılarda PHP tabanlı web uygulamaları kullanılmıştır. 2012-2015 yıllarında yapılan saldırılar incelendiğinde bu saldırıların çok odaklı olduğu özellikle finans sektörünü hedef aldığı görülmektedir[4][1].

Bölüm 3

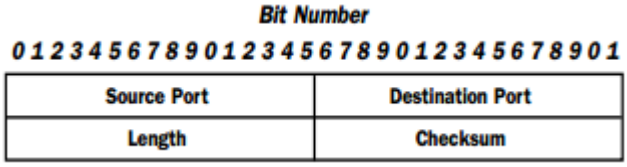
DDoS Saldırı Çeşitleri

DDoS saldırı çeşitlerini amaçlarına göre ikiye ayırmak mümkündür. Bunlar bantgenişliği doldurma ve kaynak tüketimi olarak tanımlanabilir. Bantgenişliğini doldurmaya yönelik saldırılarda hedef ağ istenmeyen paketlerle doldurulur. Bu durum normal olarak kabul edilen trafiği engeller. Kaynak tüketme saldırıları ise hedef sistemin kaynaklarını tüketmeyi amaçlar.

Kaynak tüketme saldırıları genel olarak ağ protokollerini kötüye kullanan paketleri veya bozuk paketleri içermektedir. Bu bölümde DDoS saldırı çeşitleri yapılış şekillerine göre incelenmiştir. Saldırı çeşitleri incelenirken saldırının hedef aldığı protokol de anlatılmıştır. Çünkü saldırılar çoğu zaman bir protokol zafiyetini kullanmaktadırlar.

3.1 UDP Seli

UDP (User Datagram Protocol) bir taşıma katmanı protokolüdür. UDP güvenilir olmayan bir protokoldür. Bunun nedeni gönderilen paketlerin hedefe ulaşıp ulaşmadığının kontrol edilmemesidir. Yani bu protokol ağ üzerinden paketi gönderir ama paketin gidip gitmediğini doğrulayamaz. Bu yüzden UDP protokolü boyutları küçük olan verilerin iletiminde kullanılır. UDP protokolünde iletişim sırasında bir bağlantı oluşturulmaz. Yerine ulaşmayan paketlerin tekrar gönderilmesi söz konusu değildir[5]. UDP paketleri çoğunlukla ses ve video gönderiminde kullanılır. TCP'ye göre daha hızlıdır fakat güvenlir değildir. Şekil 3.1'de UDP başlığı formatı görülmektedir;



Şekil 3.1: UDP Başlığı Formatı

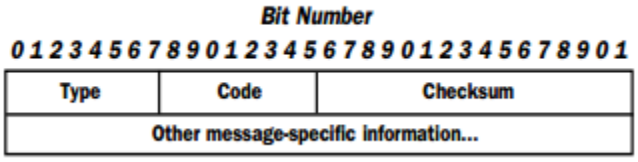
Kaynak Port (Source Port): UDP paketini gönderen tarafın kullandığı port numarasıdır.
Hedef Port (Destination Port): UDP paketini alan tarafın kullandığı port numarasıdır.
Uzunluk (Length): UDP paketinin (başlık + veri) toplam uzunluğunu verir.
Hata Sınama Bitleri (Checksum): UDP paketinin hatasız iletilip iletilmediğini belirlemek için kullanılır.

DNS, TFTP (Trivial File Transfer Protocol) ve SNMP (Simple Network Management Protocol) protokolleri UDP protokolünü kullanan protokollere örnek olarak verilebilir. UDP protokolü hızlı olduğundan ve ağ üzerinde fazla bantgenişliği kaplamadığından az miktardaki verilerin eş zamanlı olarak iletiminde çoğu zaman tercih edilir. UDP protokolü bir doğrulama mekanizması içermediğinden DDoS saldırılarına açıktır.

UDP seli saldırısında hedef sistemden bir UDP bağlantı noktası seçilir ve bu bağlantı noktasına çok sayıda UDP paketi gönderilir. Hedef sistemin bant genişliğinin ve kaynaklarının tüketilmesi amaçlanır.

3.2 ICMP Seli

ICMP (Internet Control Message Protocol) paket aktarımı sırasında meydana gelebilecek hataları elde etmek için kullanılan bir protokoldür. Sistemler arası hata kontrol mesajları bu protokol kullanılarak oluşturulur. ICMP, IP ile aynı düzeyde olmasına karşın aslında kendisi de IP'yi kullanmaktadır. ICMP protokolü, internet protokolünü daha güvenli hâle getirmeyip sadece paket iletimi sırasında meydana gelen hataların sebepleri ile ilgili bilgi sağlar[5]. Şekil 3.2'de ICMP başlığında bulunan alanlar görülmektedir;



Şekil 3.2: ICMP Başlığı Formatı

Tür (Type): ICMP mesajlarının türünü belirtir.
Kod (Code): Mesaj parametrelerini belirlemek için kullanılır.
Hata Sınama Bitleri (Checksum): ICMP paketinin tamamı üzerinden hata sınaması yapmak için kullanılır.
Mesaj (Message): Gönderilecek mesajı içerir.

ICMP paketleri kullanılarak gerçekleştirilen DDoS saldırıları ICMP seli olarak adlandırılmaktadır. Ping komutu en çok bilinen ICMP protokolü uygulamasıdır. Ping komutu ile seçilen hedef bilgisayara ICMP "echo request" paketi gönderilebilir. Eğer hedef bilgisayar bu isteğe karşı ICMP "echo response" paketi gönderir ise hedef bilgisayarın erişilebilir olduğu anlaşılır.

Ping seli saldırı çeşidinde hedef bilgisayara eş zamanlı olarak bir çok ICMP "echo request" paketi gönderilir. Eğer yeteri kadar fazla bilgisayar ile bu işlem yapılırsa hedef sistemin hizmetleri kesintiye uğratılabilir.

3.3 Parçalanmış Paket Atağı

Bu tip saldırılarda 65536 baytlık paketler hedef bilgisayara parçalara ayrılarak gönderilmektedir. Bu parçalar hedef sistemde birleştirilmeye çalışıldığında bellek taşması meydana gelmektedir. Eğer bir koruma sistemi yok ise hedef sistem servis dışı kalabilir.

3.4 Ping of Death

Ping of Death saldırısında ICMP "echo request" paketleri büyültülerek hedef sistemin bu paketlere cevap verememesi sağlanır. Genelde bu saldırı hedef bilgisayara 65535 Bayt'dan büyük IP paketleri gönderilmesiyle gerçekleştirilir. Günümüzde Ping of Death saldırısı etkinliğini kaybetmiştir. Çünkü ağ protokolleri bu büyüklükteki paketleri illegal olarak görmekte ve engellemektedir.

3.5 Slowloris

HTTP (Hypertext Transfer Protocol) istemci ve sunucu arasında çift taraflı çalışan, veri iletimini sağlayan uygulama katmanında bulunan bir protokoldür. Temelde HTTP protokolünün bir metin aktarım protokolü olduğu söylenebilir.

Bir istemci bir web sayfasına bağlanmak istediğinde o web sayfasını barındıran sunucudan bir HTTP isteğinde bulunur. Sunucu bağlantı talebinde bulunan istemci bilgisayarın yetkilerini kontrol ederek ilgili talebin cevabını istemciye gönderir. Daha sonra istemci-sunucu arasındaki bağlantı kesilir.

Sunucular kapasiteleri doğrultusunda HTTP isteklerine cevap verebilirler. Her sunucunun aynı anda açık tutabileceği bağlantı sayısı bellidir. Her istemci-sunucu bağlantısı için sunucu tarafında bir kaynak ayrılmaktadır.

Slowloris saldırısında amaç istemci-sunucu arasında gerçekleştirilen HTTP bağlantısının süresini uzatmaktır. Böylece sunucunun açık tutabileceği HTTP bağlantı sayısı azaltılmış ve sunucunun kaynakları tüketilmiş olur.

Bir sunucuya HTTP Get paketinin başlığının değiştirilmesiyle elde edilen bu istek bir çok istemci tarafından gönderilirse her bir istek için sunucu üzerinde bir kaynak ayrılacak ve sunucunun açık tutabileceği bağlantı sayısı azaltılmış olacaktır.

3.6 HTTP Get Seli

Bu saldırı çeşidinde bir çok istemciden hedef sunucuya HTTP GET isteği gönderilir. Bunun sonucunda hedeflenen, sunucunun açık tutabileceği HTTP bağlantı sayısını azaltmak ve sunucunun kaynaklarını tüketmektir. Saldırı sonucunda sunucunun HTTP Get isteklerine cevap verememesi amaçlanmaktadır. Bu saldırı türünde saldırgan hesap yükü barındıran sayfaları hedef olarak seçmektedir[6].

3.7 HTTP Post Seli

Bu saldırı türünde HTTP protokolünün POST komutu kullanılmaktadır. POST komutu bir form alanındaki içeriğin sunucu taraflı çalışan bir sayfaya aktarılması için kullanılır. Form içerisine sunucu tarafında işlenemeyecek kadar büyük veri yerleştirilerek sunucunun işlemci kaynaklarının tüketilmesi hedeflenmektedir. Bu saldırı yeteri kadar bilgisayarla gerçekleştirilir ise sunucu tarafında kaynak tükenmesine bağlı olarak hizmet kesintileri meydana getirilebilir. SSL kullanılan servislere yapılacak POST seli atakları daha fazla CPU tüketimine sebep olmaktadır[6].

3.8 Slow Post

Bu saldırı çeşidinde POST metodunda kullanılan **content-length** özelliğinin değeri büyük tutulur. Bunun sonucunda mesaj parçalara ayrılacak ve bu parçaların birleştirilmesi için geçen süre içerisinde sunucu üzerinde kaynak tüketimi olacaktır. Saldırının amacına ulaşması için bir çok bilgisayarın bir hedef bilgisayara aynı anda bu saldırıyı gerçekleştirmesi gerekmektedir.

3.9 Apache Killer

Tamamlanmamış HTTP paketleri Apache sunucuya gönderilmekte ve sunucun kaynaklarının tüketilmesi amaçlanmaktadır. Temel amaç Apache sunucunun yüksek derecede CPU ve RAM kullanması ve sonuç olarak servis veremez hale gelmesidir. Genelde bu saldırı türü Apache zaman aşımı değerinin yüksek olmasını kullanmaktadır.

3.10 DNS Seli ve DNS Sorgulama Saldırısı

Internet'in çalışmasını sağlayan ana protokollerden birisi DNS protokolüdür. Bu protokolün güvenliği diğer protokollerden daha fazla önem arz etmektedir. Bir ülkeye yönelik yapılabilecek en etkili DDoS saldırısı o ülkenin en yetkili DNS sunucularına yapılacak saldırılardır. Bu tür saldırılar sonucunda o ülkenin internet alt yapısı uzun süre hizmet dışı kalabilir.

DNS Seli saldırıları genelde iki şekilde gerçekleştirilirler. Birincisi hedef DNS sunucuya kapasitesinin üzerinde DNS paketleri göndererek gerçekleştirilmektedir. Sonuç olarak DNS sunucunun kaynaklarının tüketilmesi ve hizmet veremez hale gelmesi hedeflenmektedir. İkinci yöntemde DNS sunucusunun önüne konumlandırılmış saldırı tespit sistemlerinin kaynaklarını zorlayarak arkada bulunan tüm cihazları erişilmez hale getirmek hedeflenmektedir.

DNS sorgulama saldırısı türünde saldırgan bir DNS sunucusuna özel olarak hazırladığı bir adı sorgulatır. DNS sunucusu kendisinde olmayan bu alan adı için kök DNS sunucusundan bir istekte bulunabilir. Bu aşamada saldırganlar kaynak olarak kurban IP adresini seçerek ara sunucudan aynı alan adını defalarca sorgulatacaklardır. Hedeflenen ara DNS sunucu ile kurban bilgisayarı arasında büyük bir trafik oluşturmak ve kaynakların tüketilmesini sağlamaktır[7].

Bölüm 4

DDoS Saldırılarından Korunmak İçin Alınması Gereken Önlemler

DDoS saldırılarının temel hedefi sisteme sızmak değil sistemin verdiği hizmetleri aksatmaktır. Bu tür saldırılar da aynı anda bir çok bilgisayar hedef bilgisayara saldırır. DDoS saldırılarının tüm çeşitlerinden korunmak için tek bir yöntem yoktur. Bu bölümde açıklayacağımız yöntemler ile saldırıların etkileri ve sistemlere verdikleri zararlar azaltılabilir. Günümüzde sistemleri bu tür saldırılardan korumak için bir çok firma faaliyet göstermektedir.

Potansiyel DDoS Hedefleri;

- Yönlendiriciler, Güvenlik Duvarları
- Web Sunucuları
- DNS, Mail ve IRC Sunucuları
- Online Alışveriş Merkezleri
- Oyun ve Eğlence Merkezleri
- Veri Barındırma Merkezleri
- Aktif Ağ Cihazları

DDoS saldırıları hedef sisteme ciddi zararlar verebilmektedir. Verisign tarafından yapılmış bir değerlendirme anketinin sonucu aşağıda verilmiştir[8];

- Yanıt verenlerin %63'ü geçen sene içinde en az bir DDoS saldırısıyla karşılaştı,

- Saldırıya uğrayanların %11'i, altı veya daha fazla kez saldırıya uğradı,
- Ankete katılanların %67'si her türlü hizmet dışı kalma süresinin müşterilerini etkilediğini söyledi,
- Ankettekilerin %51'i bu tür bir hizmet dışı kalma süresinin gelir kaybına yol açtığını bildirdi.

Buna ek olarak, bazı büyük ölçekli firmaların DDoS ataklarından korunmak için, CS3 Inc. Tarafından satılan Mananet, Mazu Networks tarafından satılan PowerSecure, Arbor Network tarafından satılan Peak Flow, Asta Network tarafından satılan Vantage System gibi piyasada bulunan özel ürünleri kullandıkları görülmektedir. Türkiye'de DDoS koruması servis sağlayıcılar üzerinden alınabilmektedir.

4.1 Ağ Trafiğini İzlemek

Ağ trafiğini izlemek, sistemleri hedef alan DDoS saldırı çeşitlerini algılayabilmek için önemlidir. Bu durumda ortaya çıkabilecek eksiklikler önceden giderilebilir. Temel amaç DDoS saldırısı yapılmadan önce sistemin zafiyetlerini tespit etmek ve gidermektir.

4.2 Yük Dengeleyicileri Kullanmak

Yük dengeleyicilerini kullanmak sistemlerimizin sunduğu hizmetlerin devam etmesini sağlayan en önemli etkenlerden birisidir. Sistemde bir sunucu yerine birden fazla sunucu aktif olarak devrede olduğundan saldırı sonucunda hizmet aksaması yaşanması riski azaltılmış olur.

4.3 Saldırı Tespit ve Önleme Sistemleri Kurmak

Servis dışı bırakma saldırılarında ağ trafiğini gelişmiş bir yazılım ya da donanım kullanmadan dinlemek tek başına yeterli olmamaktadır.

Saldırı Tespit Sistemleri'nin temel amacı ağa yapılan saldırıları tespit edip kayıt altına almak ve sistem yöneticisine gerekli uyarılarda bulunarak saldırıya karşı zamanında gerekli önlemlerin alınmasını sağlamaktır. Bu sistemler; ağ trafiğini, sistem dosyalarını ve log dosyalarını izlerler. Bu izleme işlemlerini kendileri için tanımlanan kural kümelerine göre yaparlar.

Saldırı Önleme Sistemleri'nin saldırı tespitinin yanında saldırı önleme yetenekleri de bulunmaktadır. Bir saldırı önleme sistemi saldırının geldiği IP adresini bloklama özelliğine sahiptir.

4.4 Saldırıları Saptırmak

Saldırganların saldırma yöntemlerini ve erişim isteklerini tespit etmek için saldırıya hazır yalıtılmış ortamlar kullanılabilir. Bu kaynaktan elde edilen bilgiler ışığında sistemlerde bulunan açıklar kapatılabilir[9].

4.5 Yazılım Güncellemelerinin Yapılması

Sistem üzerinde kullanılan yazılımların güncel tutulması belki de en temel ve basit korunma yöntemlerinden birisidir. Bir çok sistem üreticisi belirli aralıklar ile yazılım güncellemeleri yayınlamaktadırlar. Bu yamalar takip edilmeli ve kullanılan yazılımlar güncel tutulmalıdır.

4.6 Bantgenişliğinin Arttırılması

DDoS saldırılarının bir kısmı direk bantgenişliğini hedef almaktadır. Sistemin bant genişliği tükendiğinde hizmet aksaması meydana gelecektir. Bu açıdan alınabilecek önlemlerden bir tanesi bantgenişliğini arttırmak olacaktır.

4.7 Doğru Konfigürasyonun Yapılması

Sistem üzerinde çalışan donanım parçaları ve yazılımlar için doğru konfigürasyonun yapılması gerekmektedir. Bazen saldırı tespit ya da önleme sistemlerinin doğru konfigürasyon yapılmadığı için tamamen pasif olduğu durumlarla karşılaşılabilmektedir.

4.8 Yönlendirici Seviyesinde DDoS Saldırılarından Korunmak

Yönlendiriciler üzerinde yapılacak güncellemeler ve yazılacak ACL (Access Control List) kuralları ile saldırıların etkileri azaltılabilir. ACL kuralları yazılabilmesi için saldırıya ait bilgilerin elde edilmesi gerekmektedir[10].

4.9 Güvenlik Duvarı Seviyesinde DDoS Saldırılarından Korunmak

Güvenlik duvarları ağ paketleri üzerinde önceden tanımlanmış kurallara göre denetleme işlemi yapan cihazlardır. Bu işlemleri kaynak-hedef IP adresleri kaynak-hedef port numaraları vb. gibi bilgilere göre yapmaktadırlar. Kurumlar çoğunlukla ağ alt yapılarına giriş ve çıkış işlemlerinin tek bir yerden yapılması için yönlendiricilerden sonra güvenlik duvarları kullanılırlar.

Güvenlik duvarı; donanımla gerçekleştirilmiş bir aktif cihaz olabileceği gibi bir bilgisayara yüklenmiş bir yazılım da olabilir. Güvenlik duvarları temel olarak aşağıdaki işlemleri yapabilmektedirler[11];

- Kullanıcı tanımlı sınırlandırmalar
- Dış ve iç erişimlerin gözlenmesi istatistiklerinin tutulması
- Ağın dışarıdan izlenebilmesi ve adres dönüşümü
- Sanal ağ oluşturulması ve şifreleme işlemleri
- Erişim kısıtlamaları

DDoS saldırılarından korunmak için güvenlik duvarı üzerinde bandgenişliği sınırlandırma özelliği aktif yapılabilir. Bununla beraber her aktif cihazda olduğu gibi güvenlik duvarları ile beraber gelen ön tanımlı zaman aşımı değerleri bulunmaktadır. Bu değerlerin sınırlandırılması saldırıların etkisini azaltacaktır.

Bölüm 5

TCP SYN Seli Saldırısı ve Korunma Yöntemleri

5.1 TCP Protokolü İle İlgili Temel Bilgiler

TCP (Transmission Control Protocol) bir ulaşım katmanı protokolü olup TCP/IP protokol ailesinin en önemli üyesidir. TCP protokolünde veri transferi başlamadan önce bir bağlantı kurulur. Bu tip yapılar bağlantılı (connection-oriented) iletişim olarak adlandırılmaktadır. Oturum açılırken istemci ve sunucu kendi iletişim parametrelerini birbirlerine aktarırlar. Veri iletişimi önceden belirlenen bu parametrelere göre yapılmaktadır. TCP protokolü çift yönlü çalışan bir protokoldür. Veri gönderimi ve alımı aynı anda yapılabilmektedir. Bu yapı Çift Yönlü İşlem (Full Dublex Process) olarak adlandırılır ve bağlantıyı hızlandırır.

Üst katmandan gelen veriler uygun uzunlukta parçalara ayrılır. Her bir parça bölüm (segment) olarak adlandırılır. Her bir segment; TCP başlığı ve veri olmak üzere iki alandan oluşmaktadır. Alıcı tarafında bu parçaların doğru bir şekilde sıralanması için her bir segment bir sıra numarasına sahiptir. Alıcı tarafında segmentler bu sıra numaralarına göre birleştirilerek veri yükünün tamamı elde edilir.

TCP protokolünde veri iletimi güvenilir bir şekilde yapılmaktadır. Paketlerin karşı tarafa ulaşıp ulaşmadığı ACK (Acknowledgment) bilgisi ve hata denetim özellikleri ile kontrol edilir. Eğer ACK mesajı doğru bir şekilde gelmezse yani paket bozulmuş ya da kaybolmuş ise bu durumda ilgili paket alıcıya tekrar gönderilir. Özetle bilgilerin hedef sisteme gidip gitmediği kontrol edilir. TCP protokolü akış kontrolü mekanizması içermektedir.

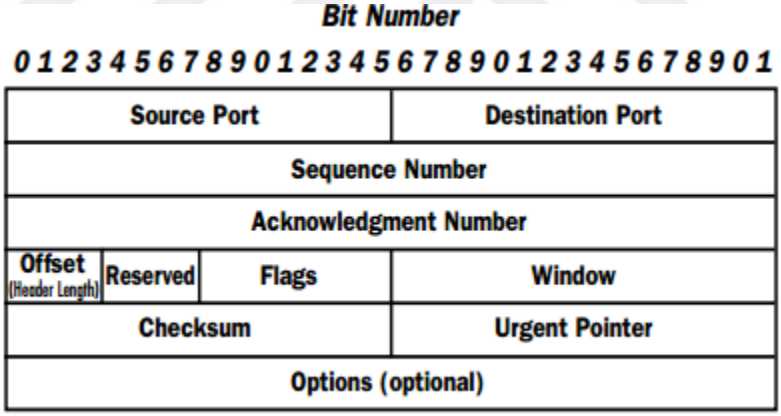
TCP protokolünde paketlere gönderici ve alıcı port numaraları eklenmektedir. Bu durum uygulamalar arasında bir bağlantı kurulduğu anlamına gelir. Kaynak port

numarası bilgisi TCP isteğinde bulunan uygulamanın port numarasını belirtir. Hedef port numarası ise gönderilen bilginin alıcı tarafında hangi servis tarafından kullanılacağını gösterir.

TCP protokolü ile ilgili anlatılanların özeti aşağıda çıkartılmıştır[5];

- TCP bağlantı yönelimli (connection-oriented) çalışan bir protokoldür.
- Veriler parçalara bölünerek yollanır. Her bir parça segment olarak adlandırılır.
- Çift yönlüdür, aynı anda çift taraflı veri iletişimi yapılabilir.
- Veri iletimi güvenilir bir şekilde yapılmaktadır.
- Segmentlerin alıcı tarafında doğru olarak sıralanabilmesi için her bir segment sıra numarası içerir.
- Akış kontrolü yapılmaktadır.

TCP protokol başlığının incelenmesi[11][12];



Şekil 5.1: TCP Protokol Başlığı

Kaynak Port (Source Port): TCP oturumu için göndericinin kullandığı 16 bitlik port numarasını saklar. Bir TCP oturumunda bilginin tüm alt parçaları aynı gönderici port numarasına sahip olurlar. Eğer kullanılan uygulamada veriyi gönderen tarafın bağlantı noktası bilgisi gerekli değil ise bu alanda saklanan değer 0 olacaktır.
Hedef Port (Destination Port): TCP oturumu için alıcının kullandığı 16 bitlik port numarasını saklar. Alıcı cihazın port numarası bilinmeden veriler hedefe ulaşamaz.
Sıra Numarası (Squence Number): Tüm veri içerisinde TCP segment'inin birinci baytının sıra numarasını gösterir.
Onaylama Numarası (Acknowledgment Number): ACK bitinin 1 olması

onaylama numarası alanının geçerli olduğu anlamına gelmektedir. Alıcının hangi sıradaki segmen'ti beklediği onay numarası ile belirlenir.

Başlık Uzunluğu (Data Offset,Header Length): TCP segmenti içerisinde başlık bilgisinin uzunluğunu belirtir. Buradaki sayı 4 ile çarpılarak başlık uzunluğu elde edilir. Örneğin başlık 32 bitlik seçimlik alanı içermiyorsa bu alanda 5 değeri saklanacaktır. Bu durumda başlık toplam 20 bayt olur. Bu alanın kullanım amacı seçimlik alanının değişken uzunluklarda olabilmesidir.

Ayrılmış Alan (Reserved): Saklı tutulmuş alandır. Bu alanlarda 0 değeri saklanır.

TCP Bayrakları (Flags): TCP bağlantıları bayraklarla yürütülür. Bayraklar bağlantıyı kurma, veri transferi, onay mekanizması, bağlantıyı sonlandırma, denetim bilgilerini düzenlemek için kullanılır. Diğer bir değişle bayraklar bağlantının durumunu belirleme işlemine yaramaktadır. Aşağıda bayraklar listelenmiştir.

CWR (Congestion Window Reduced, Tıkanıklık Penceresi Azaltma): Tıkanıklık kontrol mekanizmasının aktif hale getirilmesi için kullanılır.

ECE (Explicit Congestion Notification Echo, Açıkça Tıkanıklık Belirtisi): Eğer bir tıkanıklık oluşmuş ise TCP paketinin korunması gerektiğini bildirir. Genel olarak CWR ve ECE bitleri yönlendiricilerde ortaya çıkabilecek çakışmaları engellemek için kullanılan kontrol bitleridir.

URG (Urgent, Acil): Bu bitin değeri 1 ise; gönderilen veri önemli ve paket içerisindeki acil göstergesi (urgent pointer) alanı kullanılmış demektir.

ACK (Acknowledgment Number): Bu bitin değerinin 1 olması onay numarası alanının kullanıldığı diğer bir ifade ile geçerli olduğu anlamına gelir.

PSH (Push): Bu bit değeri ile paketin hemen işleme sokulması sağlanabilir. Eğer PSH değeri 1 ise paket hemen işleme sokulacaktır.

RST (Reset): Eğer bu bitin değeri 1 ise TCP bağlantısı başlangıç durumuna geri döndürülür.

SYN (Synchronize): Bu bit TCP oturumunu başlatmak için kullanılır. TCP bağlantısının başlaması için gönderilen ilk pakette SYN bayrağı 1 olarak ayarlanır.

FIN (Finish): Bu bit "1" olarak ayarlandığında paketi gönderen tarafın TCP bağlantısını kapatmak istediği hedef bilgisayar tarafından anlaşılır. Bilgi gönderimi artık tamamlanmıştır.

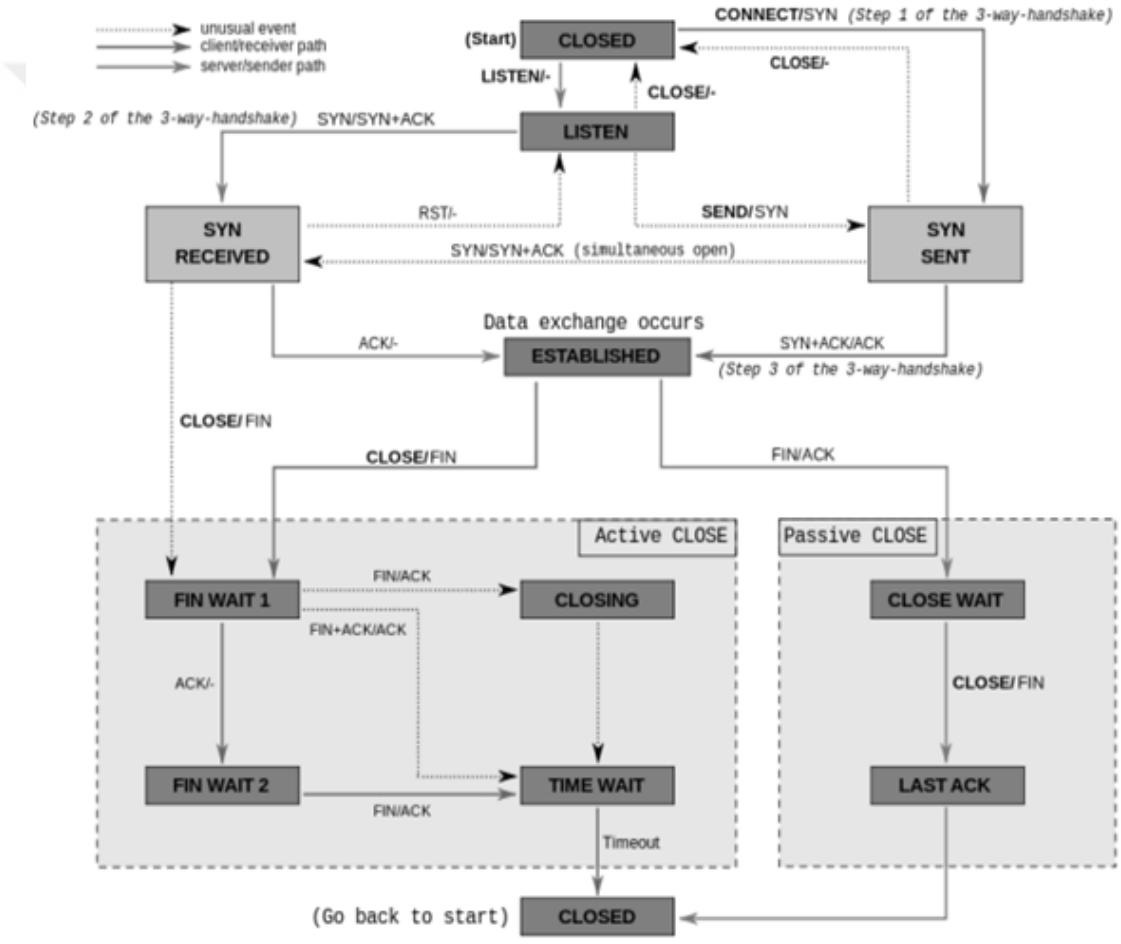
Pencere Genişliği (Window Size): Tampon bellekte kullanılabilir alanın bayt cinsinden uzunluğunu gösterir.

Hata Sınama Bitleri (Checksum): Tüm TCP segmenti üzerinden hesaplanan hata kontrolü işlemleri için kullanılır. TCP veri başlığı ve verisi üzerinden hata kontrolü yapılır. Eğer alıcı tarafından hesaplanan bu değer göndericinin hesapladığı değerle aynı değil ise segment devre dışı bırakılır.

Acil İşaretçi (Urgent Pointer): Bu alan URG bayrağı ile beraber acil veri aktarımı için kullanılır.
Seçimlik Alan (Option): TCP başlığı içerisinde ekstra bilgi göndermek için kullanılan alandır. Bu alan 32 bitin katları şeklinde kullanılmalıdır. Eğer alan 32 bitin katı şeklinde değil ise boşluk doldurma (padding) bitleri ile 32 bitin katı haline getirilir.

TCP bağlantı noktası durumları aşağıda verilmiştir[5].

CLOSE_WAIT, CLOSED, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, LAST_ACK,LISTEN, SYN_RECEIVED, SYN_SEND, TIME_WAIT şeklindedir.



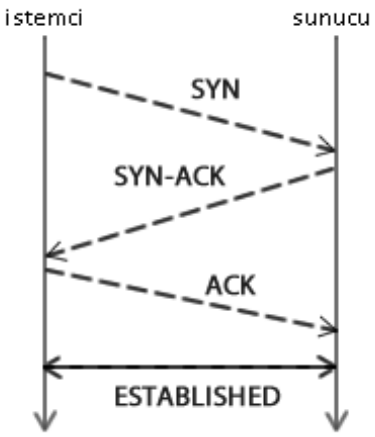
Şekil 5.2: TCP Durum Diyagramı

LISTEN: Portun bağlantı kabul ettiğini belirtir.
SYN_SEND: İstemci, sunucu arasında TCP bağlantısı oluşturmanın ilk adımıdır. SYN bayraklı paket gönderilmiş ve buna karşın cevap beklenmektedir.
SYN_RECEIVED: Hedef sistem SYN paketini almış ve cevap olarak SYN-ACK paketi dönmüştür. Hedef sistem ACK paketi beklemektedir.

ESTABLISHED: Üç yollu el sıkışma tamamlanmıştır. Artık istemci ve sunucu veri transferi yapabilecek durumdadır.
FIN_WAIT_1: Bağlantıyı sonlandırmak için işlem başlatan taraf (x) hedef sisteme FIN bayraklı bir TCP paketi gönderir. Karşı taraftan (y) ACK ve FIN bayraklı iki paket beklemeye başlar. Bu arada kendi port durumunu FIN_WAIT_1 olarak ayarlar.
FIN_WAIT_2: (y) tarafı FIN bayraklı paketi aldıktan sonra (x) tarafına ACK bayraklı bir paket gönderir. Bu durumda (y) tarafı CLOSE_WAIT durumuna, (x) tarafı FIN_WAIT_2 durumuna geçecektir. Bu aşamada (x) tarafı (y) tarafından FIN bayraklı paket beklemektedir.
CLOSE_WAIT: (x) tarafı FIN_WAIT_2 durumuna geçtiğinde (y) tarafı CLOSE_WAIT durumuna geçmektedir.
LAST_ACK: (y) tarafı (x) tarafına FIN bayraklı paket gönderdiğinde kendisini LAST_ACK durumuna geçirir.
TIME_WAIT: (x) tarafı (y) tarafından gönderilen FIN bayraklı paketi almıştır ve (y) tarafına ACK bayraklı bir paket göndermiştir. Bu durumda (x) tarafı TIME_WAIT durumundadır.
CLOSED: Bağlantının sonladığı durumdur.

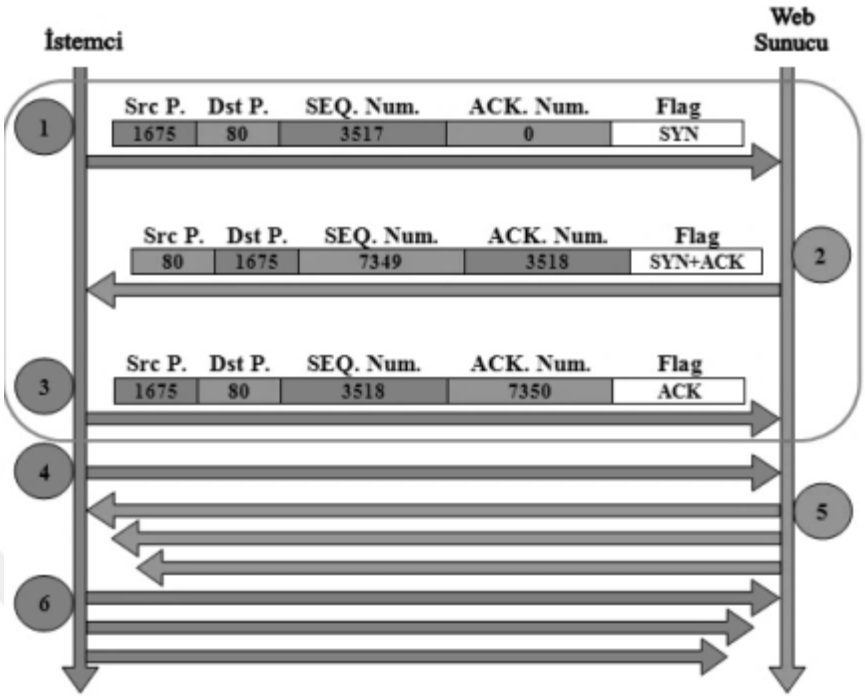
5.2 TCP Oturum Başlatma ve Sonlandırma

SYN paketleri sunucu ve istemci arasındaki haberleşmeyi başlatmak amacıyla kullanılır. Bir istemci sunucuya bağlanmak istediğinde sunucuya ağ üzerinden SYN paketi gönderir. Sunucu bu paketi SYN-ACK paketi ile yanıtlar ve son olarak istemci sunucuya ACK paketi gönderdiğinde haberleşme başlamış olur. Bu iletişime Üç Yollu El Sıkışma (Three Way Handshake) denilmektedir.



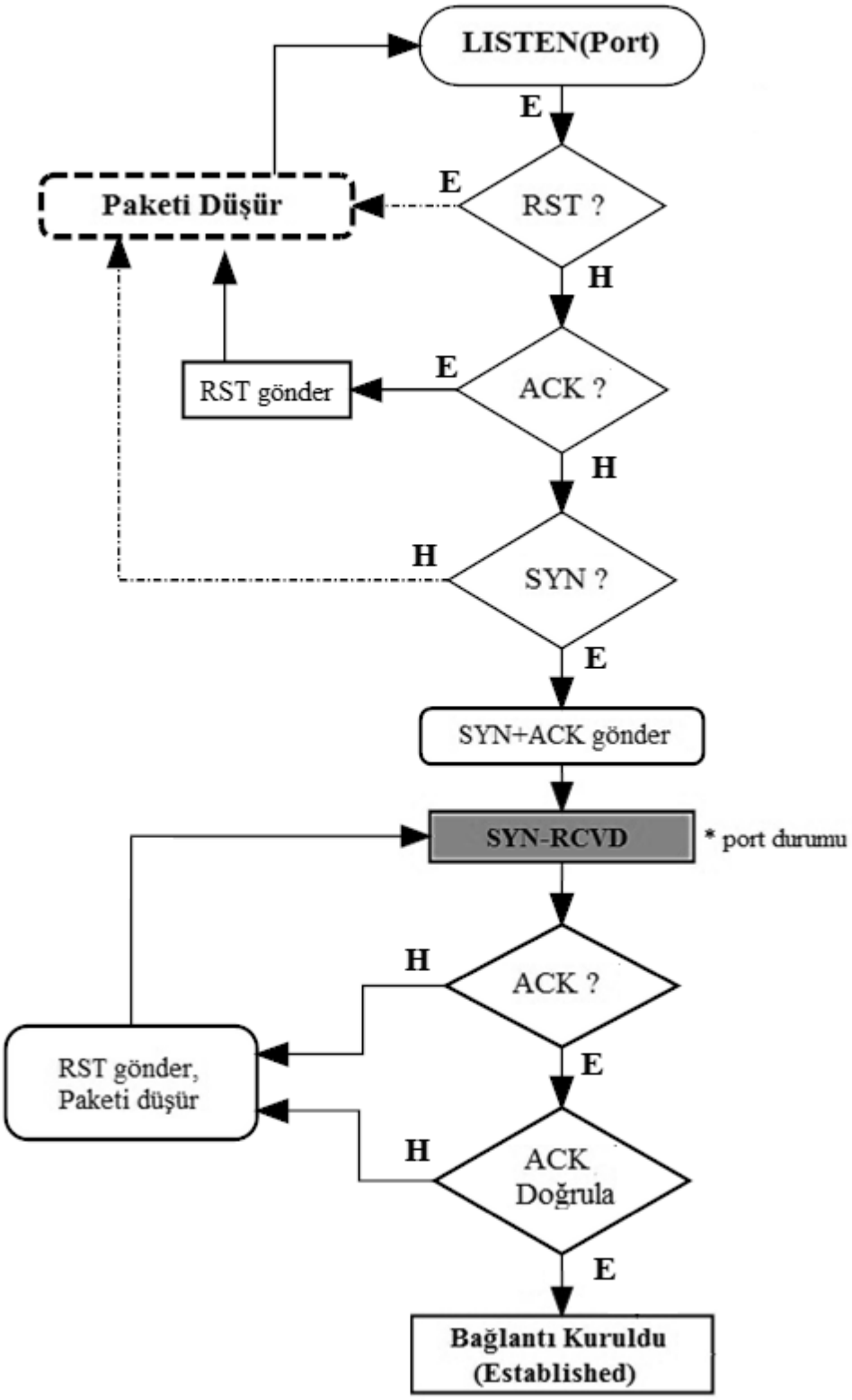
Şekil 5.3: Üç Yollu El Sıkışma

Established durumunda artık veri transferi başlamış olmaktadır. Üç yollu el sıkışma'nın nasıl gerçekleştiği Şekil 5.4'de daha ayrıntılı bir şekilde incelenmiştir.



Şekil 5.4: Üç Yollu El Sıkışma (Ayrıntılı)

İstemciden sunucuya ilk SYN paketi geldiğinde sunucu bu bağlantı için bellekten bir alan ayıracaktır. Bunun yanında sunucu ilk SYN paketine karşılık gönderdiği SYN-ACK paketinde SYN değerini rastgele olarak atamaktadır. İlk SYN değeri bazı kaynaklarda ilk sıra numarası (initial sequence number) olarak adlandırılmaktadır. 2.Pakette ACK değeri 1.pakette gelen SYN değerinin bir fazlasıdır. 3.Pakette sunucuya gelen ACK değeri 2.paketin SYN değerinden bir fazla ise bağlantı başlar. Aksi durumda bağlantı başlamayacaktır. Aşağıda TCP bağlantısının kurulması akış şeması formatında gösterilmiştir[13] [14].



Şekil 5.5: TCP Bağlantısının Kurulması

Bir TCP paketi LISTEN modda olan bir porta ulaştığında ilk olarak RST bayrağı ikinci olarak ACK bayrağı denetlenmektedir. Eğer RST bayraklı bir TCP paketi

LISTEN modda olan bir porta gelirse ilgili paket düşürülür. Diğer yandan ACK bayraklı bir paket bağlantı noktasına ulaştığında ilk önce bu paketi gönderen bilgisayara RST paketi gönderilir ve daha sonra ilgili paket düşürülür.

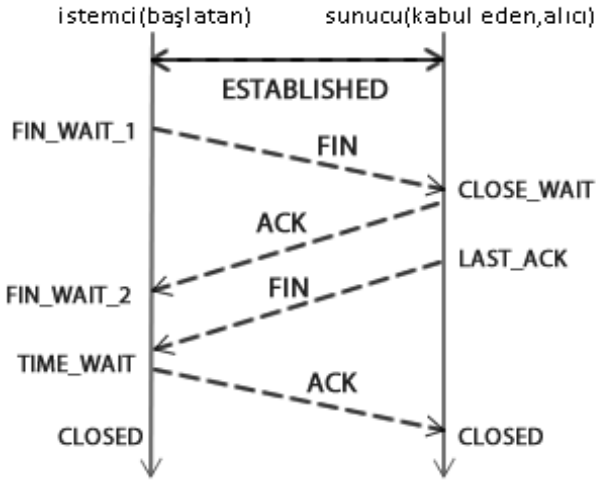
RST ve ACK bayraklarından sonra SYN bayrağı denetlenmektedir. Eğer SYN bayrağı işaretlenmiş ise bu durumda sunucu istemciye SYN-ACK paketi gönderecek ve kendi portunu SYN_RECEIVED durumuna çekecektir. Sunucu bu bağlantı için artık SYN geliş zamanını, kaynak ve hedef port bilgilerini, sıra numaralarını ve istemciye ait diğer bilgileri depolayabilir durumdadır.

Sunucu bu bağlantı için üç yollu el sıkışma tamamlanana kadar bellekte bir alan ayırır. Bu alanda bağlantıyla alakalı bilgiler saklanmaktadır. SYN_RECEIVED durumunda olan bağlantılar yarı açık ya da kısmi açık bağlantılar olarak adlandırılmaktadır. SYN_RECEIVED durumundaki yarı açık bağlantıların tamamı; birikim/bekleme listesi (backlog queue) ya da SYN listesi (SYN queue) olarak adlandırılmaktadır[13].

Bir sonraki aşamada istemciden uygun ACK yanıtı beklenmektedir. Bu durumda sunucu ACK bayraklı bir paket dışında başka bir paket alırsa, gelen paket düşürülür ve istemciye RST paketi gönderilir. Sunucu kendi durumunu tekrar SYN_RECEIVED olarak günceller.

Bu aşamada eğer ACK bayraklı bir paket sunucuya ulaşmış ise bir sonraki adımda TCP başlığında bulunan onaylama numarası'nın doğru olup olmadığı test edilir. Eğer onaylama numarası doğru değil ise gelen paket düşürülür ve istemciye RST paketi gönderilir. Sunucu bağlantı noktası durumunu SYN_RECEIVED olarak günceller. Eğer ACK doğru ise sunucu ve istemci arasında veri transferi başlayacaktır.

Oturum sonlandırma işlemi 4 adımda gerçekleştirilmektedir. Oturum sonlandırma işleminde her iki taraf aktif durumdadır.



Şekil 5.6: TCP Oturum Sonlandırma

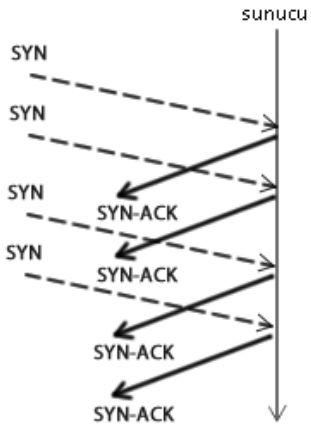
Oturum sonlandırma işlemi FIN bayraklı paketin hedef bilgisayara gönderilmesiyle başlamaktadır. Hedef bilgisayar FIN paketine karşılık ACK ve FIN paketleri ile karşılık vermektedir. Son olarak ACK bayraklı paketin hedef bilgisayara gönderilmesiyle oturum sonlandırılmış olur.

5.3 TCP SYN Seli Saldırısı

SYN seli saldırısı, TCP sunucusuna karşı yapılan servis dışı bırakma saldırısıdır. 1994 yılında ilk bilinen SYN saldırısını yapan kişi Kevin Mitnick'dir. Mitnick bu yöntemle ağdaki sunucuyu devre dışı bırakarak kendi sahte saldırgan sunucusunun yönetici olarak atanmasını sağlamıştır[15].

SYN seli saldırısında amaç sunucuya çok fazla SYN paketi gönderip sunucunun sistem kaynaklarını tüketmek ve sunucuyu hizmet veremez duruma getirmektir. Bir çok noktadan bir sunucuya SYN paketi gönderildiğinde sunucu bu paketlere SYN-ACK paketleri ile karşılık verecek ve kendi port durumunu SYN_RECEIVED olarak ayarlayacaktır. Normal şartlarda atak yapmayan istemciler ACK paketlerini hemen gönderdiklerinden sunucu SYN_RECEIVED durumundan sadece milisaniyeler içinde çıkmaktadır. Bu aşamadan sonra eğer saldırgan durumunda olan bilgisayarlar sunucuya ACK paketleri göndermezler ise sunucu SYN_RECEIVED durumunda kalacak ve sunucu-saldırgan(lar) arasındaki haberleşme başlamayacaktır.

Bu durumda SYN listesi maksimum seviyeye ulaşacak ve yeni bağlantılar kabul edilmeyecektir. Sonuç olarak sunucu, saldırı yapmayan normal istemcilere cevap veremez hale gelecektir. SYN seli saldırılarının başarılı olabilmeleri için TCP bağlantı noktasının açık ve LISTEN modda olması gerekmektedir. Eğer TCP bağlantı noktası kapalı ise RST dönecektir.



Şekil 5.7: SYN Seli Saldırısı

Buradaki problem sunucu üzerinde her bir bağlantı için ayrılan bellek alanıdır. Bu bellek alanları belirli bir süreliğine ayrılmış olsalar bile sunucu hizmetinin aksamasına sebep olabilirler. Bu tür bağlantılar sunucu kaynaklarını tüketen bağlantılardır.

Aşağıda SYN seli saldırısı ayrıntılı bir şekilde incelenmiştir;

1. Öncelikle saldırgan/saldırganlar bir araç kullanarak sahte IP adreslerini içeren bir çok SYN paketi üretir ve kurban sunucuya gönderir.

2. Sunucu her bir SYN paketi için bir SYN-ACK paketini sahte IP adreslerine gönderir ve her bir yarı-açık bağlantı için bellekte bir alan ayırır.

3. Sahte istemciler SYN paketi göndermediklerinden ACK paketi de göndermeyeceklerdir. Sonuç olarak hiçbir zaman sahte istemciler ve sunucu arasında bağlantı kurulmayacaktır.

   Bu aşamada iki durum söz konusudur.

   - **Sahte İstemcideki Port Güvenlik Duvarı Korumasız İse;** SYN-ACK paketi sahte istemcideki LISTEN ya da CLOSE durumunda olan bağlantı noktasına ulaşır. Bağlantı noktası geriye RST paketi gönderecektir. Bu durumda ilgili bağlantı için sunucunun ayırdığı bellek serbest bırakılır.
   - **Sahte İstemcideki Port Güvenlik Duvarı Korumalı İse;** Bu durumda sunucuya ulaşacak yanıt güvenlik duvarına bağlı olacaktır. Varsayılan olarak güvenlik duvarları bağlantı noktalarını korumak amacıyla TCP paketlerini düşürür. Bu durumda sahte IP adresine sahip sistemden sunucuya hiçbir cevap dönmeyecektir. İlgili bağlantı için ayrılan bellek alanı, tanımlanan zaman aşımına kadar bellekte tutulacaktır. Yanıtın olmaması SYN seli saldırısını daha etkili bir hale getirmektedir.

4. Eğer sahte istemci/istemciler mevcut değil ise yanıt ICMP host-unreachable olarak saldırı altındaki sunucuya iletilir. Böylece ilgili bağlantı için sunucunun ayırdığı bellek alanı serbest bırakılır[16].

5. Saldırılan hedef sunucu SYN-ACK paketlerine cevap alamadığı için SYN listesi maksimum seviyeye ulaşana kadar SYN paketi almaya devam edecektir.

5.4 TCP SYN Seli Saldırısından Korunmak İçin Kullanılan Mevcut Yöntemler

Bölüm 4'de anlatılan önlemlerin yanında aşağıda TCP SYN seli saldırısına karşı alınabilecek özel önlemler incelenmiştir. Windows, Linux/Unix işletim sistemlerinde netstat komutu kullanarak SYN seli saldırısı anlaşılabilmektedir.
Aşağıda **netstat -ant|grep SYN_RECV** komutuna ait çıktıdan bir parça görülmektedir.

```
tcp        0      0 192.168.143.128:80      249.146.57.15:5710      SYN_RECV
tcp        0      0 192.168.143.128:80      254.52.115.208:61875    SYN_RECV
tcp        0      0 192.168.143.128:80      242.48.126.57:39922     SYN_RECV
tcp        0      0 192.168.143.128:80      242.158.92.125:5620     SYN_RECV
tcp        0      0 192.168.143.128:80      249.8.116.158:44087     SYN_RECV
tcp        0      0 192.168.143.128:80      242.151.243.84:5550     SYN_RECV
tcp        0      0 192.168.143.128:80      248.211.178.151:43931   SYN_RECV
tcp        0      0 192.168.143.128:80      244.16.115.175:47726    SYN_RECV
tcp        0      0 192.168.143.128:80      254.61.221.59:58267     SYN_RECV
```

Şekil 5.8: SYN Seli Saldırısı Tespiti

Şekil 5.8'de TCP bağlantılarının SYN_RECEIVED durumunda olduğu görülmektedir. Bu durum şöyle özetlenebilir; istemci sunucuya SYN paketini göndermiş ve sunucuda istemciye SYN-ACK paketini göndermiştir. Sunucu, istemciden ACK paketini alana kadar SYN_RECEIVED durumunda beklemektedir. Bu durum tipik bir SYN saldırısı durumudur.

SYN seli saldırısı olup olmadığını anlamak için diğer bir yöntem güvenlik cihazlarının durum ve bağlantı tablolarına bakmaktır. Sunucular, istemcilere hizmet vermek için TCP bağlantı isteklerini kabul ederler. Hangi bağlantının cevapsız bir bağlantı olacağı önceden kestirilemez. Dolayısıyla tüm TCP bağlantı isteklerini durduran bir güvenlik duvarı ile SYN seli saldırısının engellenmesi mümkün değildir.

Zaman aşımı değerleri varsayılan olarak çok yüksektir. Bu değerlerin düşürülmesi saldırının etkisini azaltmaktadır. Cevapsız SYN paketleri (ACK göndermeyen) için ayrılan bellek alanı zaman aşımı süresi boyunca bellekte tutulmaktadır. Bu durumda yarı açık bağlantılarda bekleme süresi ne kadar küçük olursa SYN seli saldırısının etkisi o kadar azalmış olacaktır.

Sunucu işletim sistemleri SYN bayraklı paketleri aldıklarında bu paketlere karşılık SYN-ACK bayraklı paketlerle cevap dönerler. Eğer istemcilerden ACK bayraklı paketler gelmez ise sunucu belirli bir sayıda SYN-ACK paketini istemcilere göndermeye devam eder. Bu değer tcp_synack _retries değişkeni ile belirlenir ve varsayılan olarak Linux dağıtımlarında "5" 'dir. Bu değer istemciden gelen SYN paketine karşılık kaç tane SYN-ACK paketi gönderileceğini tanımlamaktadır[17]. Bu değeri azaltmak için

net.ipv4.tcp _synack_ retries değişkenine etc/sysctl.conf dosyası içerisinde bir atama yapılabilir.

SYN seli saldırısından korunmak için kullanılan temel yöntemler SYN önbellek (cache) , SYN vekil sunucu/güvenlik duvarı ve SYN çerezleri yöntemleridir.

5.4.1 SYN Önbellek

FreeBSD sistemler ile beraber gelen SYN önbellek (cache) özelliğinde her SYN bağlantısı için bellekten daha az alan ayrılması hedeflenmektedir. Yoğun saldırılar altında bu özelliğinin işe yaramadığı görülmektedir[17].

5.4.2 SYN Vekil Sunucu/Güvenlik Duvarı

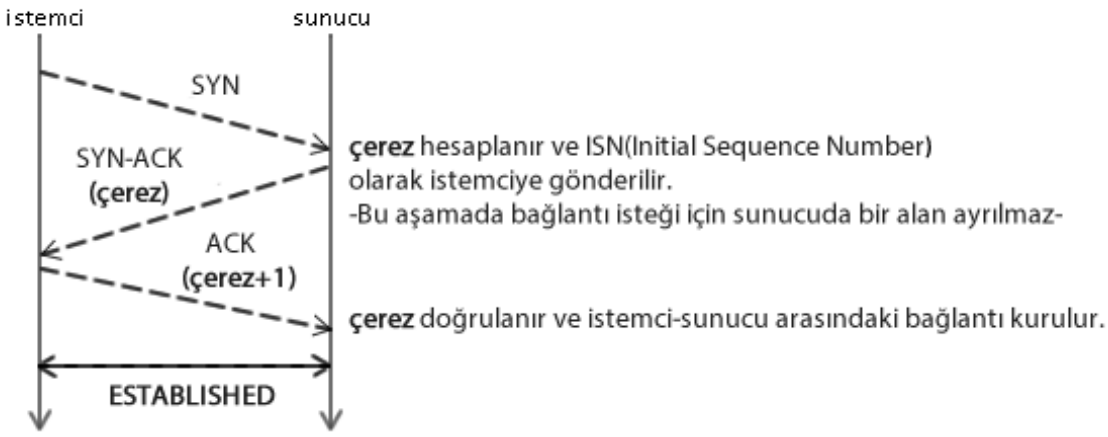
Bu yöntemde istemci ve sunucu arasındaki üç yollu el sıkışma işlemi, (istemci)-(SYN vekil sunucu/güvenlik duvarı) arasında gerçekleştirilir. Böylece hedef sunucu cevapsız SYN saldırısından korunmuş olacaktır. Eğer üç yollu el sıkışma başarılı olursa, aynı işlem (SYN vekil sunucu/güvenlik duvarı)-(sunucu) arasında sahte olarak yapılacak ve istemci-sunucu arasındaki veri transferi araya konan ağ cihazı üzerinden başlayacaktır.

5.4.3 SYN Çerezleri

SYN çerezleri, TCP SYN seli saldırısının etkisini azaltmak için kullanılan en temel yöntemlerden birisidir. SYN seli saldırısına çözüm bulmak amacıyla 1996 yılında D.J.Bernstein tarafından önerilmiştir[3]. Bu yöntem TCP için ek bir özellik olarak düşünülebilir.

SYN seli saldırısında anahtar nokta, hedef sunucunun saklayabileceği yarı açık bağlantı sayısının doldurulmasıdır. SYN çerezleri yöntemi ile SYN listesi dolu olsa bile kurban işletim sistemi TCP bağlantı isteği kabul edebilir. SYN çerezleri yöntemi SYN sırası dolduğunda devreye girmektedir[18].

SYN çezleri özelliği aktif edilmiş bir sistemde SYN sırası dolduğunda sunucu işletim sistemi, istemciden gelen SYN paketinden sonra bir kaynak ayırmaz. Bunun yerine istemciden gelen SYN paketinden istemci ile ilgili parametreler elde edilir ve SYN paketine karşılık gönderilecek SYN-ACK paketindeki sıra numarası (sequence number) değeri özel olarak hesaplanır. Bu değer sunucunun ürettiği ilk sıra numarası olduğundan Initial Sequence Number (ISN) olarak adlandırılmaktadır. Hesaplanan bu değere çerez denmektedir.

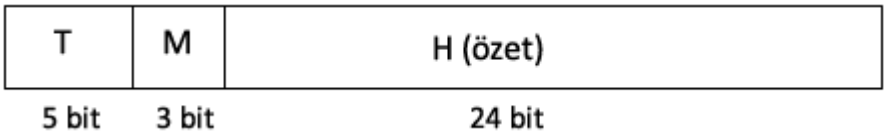


Şekil 5.9: SYN Çerezleri İle Üç Yollu El Sıkışma

İstemciden ACK paketi geldikten sonra 2.pakette gönderilen ISN (çerez) değeri tekrar hesaplanarak gelen ACK değerinin bu değerden bir fazla olması beklenir. Eğer anlatılan şartlar sağlanmış ise bağlantı oluşturulur ve veri akışı başlar. Linux sistemlerde SYN çerezleri özelliğini aktif yapmak için; /etc/sysctl.conf dosyası içerisinde aşağıdaki tanımlama yapılmış olmalıdır.

net.ipv4.tcp_SYN cookies= 1

Sunucu tarafından ISN değeri (çerez) aşağıdaki gibi hesaplanır.



Şekil 5.10: İlk Sıra Numarası Bitsel Şablonu

T: t mod 32 , burada t her 64 saniyede bir artan 32 bitlik bir zaman sayacıdır[18]. Güncel olmayan ve sahte çerezleri reddetmek ve çerez değerlerinin zamanla hızlı bir şekilde artmasını sağlamak için kullanılır. Linux uygulamalarında sadece son 4 dakikaya ait çerezler geçerlidir.

M: MSS (Maximum Segment Size) istemci ve sunucu arasında yapılan veri transferinde gönderilebilecek en büyük segment boyutudur. Kurulan her bir bağlantı için tanımlanır. M alanında istemci-sunucu arasında kullanılabilecek MSS değerine karşılık gelen 3 bitlik kodlanmış değerler saklanır. MSS değeri parçalanmaya neden olmadan olabilecek en büyük segmenti kullanmak açısından önemli bir unsurdur, normalde bağlantıyı oluşturmak için sadece ilk el sıkışma anında kullanılır.

Syn Çerez Verisi	0	1	2	3	4	5	6	7
MSS	64	512	536	1024	1440	1460	4312	8960

Yukarıdaki tabloda MSS değerlerine karşılık gelen 3 bitlik çerez verisi değerleri görülmektedir. Tabloda yer almayan MSS değerleri için; bu değerlerden bir küçük en büyük değer seçilir.

H: Aşağıdaki formüle göre hesaplanmış 24 bitlik bir değerdir[19][18].

H1=Özet(Saddr | Sport | Daddr | Dport | K1)
H2=Özet(Saddr | Sport | Daddr | Dport | counter | K2)
H= H1 + ISNclient + (counter « 24) + H2 %(1«24)[18]

D.J.Bernstein K1=0,K2=1 (K1,K2 gizli anahtarlar) olarak kullanılabileceğini ve bu durumun çerez için güvenlik kaybına neden olmayacağını belirtmiştir[18]. İkinci özet değeri (H2) zaman unsurunu gizlemek için kullanılmıştır ve formülde iki özet hesabının olması en iyi seçenektir. Andi Kleen tarafından yazılan SYN çerezleri gerçeklemesinde Single Block SHA1 algoritması kullanılmıştır. Yukarıdaki formül sonucunda elde edilen H değerinin rastgele olarak seçilen 24 bitlik bir bloğu çıktı olarak kullanılmaktadır.

Formülde kullanılan parametreler[19];

Saddr,Sport: Kaynak IP/Port bilgisi
Daddr , Dport: Hedef IP/Port bilgisi
K1, K2: Gizli anahtarlar (K1=0 , K2=1 olarak seçilebilir.)
counter: Dakika (zaman) sayacı
ISNclient: İstemci ISN değeri

Linux için SYN çerezleri gerçeklemesini yapan Andi Kleen H değeri hesaplanırken kullanılan yukarıdaki formüle aşağıdaki gibi **data** isimli bir parametre eklemiştir.

H= H1 + ISNclient + (counter « 24) + (H2 + data) %(1«24)[20]

data parametresi çerezi oluşturan 3 bitlik MSS değeridir. Yukarıdaki formülün ürettiği çerez değerleri mevcut SYN numaraları gibi zamanla artış göstermektedir. Çerez üretme yöntemi zamana bağımlıdır. Zaman bağımlılığı saldırganların başka bilgisayarlardan geçerli çerez değerleri bulmalarını zorlaştırır ama tamamen ortadan kaldırmaz[3].

2.6.32.69 çekirdek sürümü için SYN çerezleri gerçeklemesi *https://git.kernel.org /cgit/linux/kernel/git/stable/linux-stable.git/tree/net/ipv4/syncookies.c?id=refs/tags /v2.6.32.69* adresinden elde edilebilir.

5.5 SYN Çerezleri Yönteminin İncelenmesi

Bu başlık altında ilk önce SYN çerezleri hakkında literatüre geçen eleştiriler ve bu eleştirilere verilebilecek cevaplar incelenmiştir. Özellikle Alexey Kuznetsov, Wichert Akkerman ve Perry Metzger SYN çerezlerine karşı aşağıdaki eleştirileri sunmuşlardır[3].

D.J.Bernstein'in geliştirdiği SYN çerezleri yöntemine yönelik eleştirilerden birisi SYN çerezlerinin TCP Protokolünü ciddi şekilde ihlal ettiği yönündedir. Bu eleştiriye karşılık D.J.Bernstein geliştirdiği yöntemin TCP protokolü ile tamamen uyumlu olduğunu ve SYN çerezleri özelliği aktif edilmiş bir sunucunun ürettiği çerezin başka bir sunucu tarafında da üretilebileceğini belirtmektedir[3]. SYN çerezleri yöntemi TCP protokol başlığına yeni bir alan eklememektedir. Bunun yanında üç yollu el sıkışma işleminin temel mantığını da değiştirmez. Önceden tamamen rastgele sayılar kullanılarak güvenilirliği sağlamak amacıyla konulmuş olan SYN ve ACK alanları belirli bir algoritma akışı ile güvenlik amaçlı olarak kullanılamaya başlanmıştır.

TCP başlığını değiştirmeyen bu yaklaşım çok büyük güvenlik sağlamakta olup birçok işletim sisteminde uzun yıllardan beri kullanılmaktadır. Hatta literatürde yöntemi geliştirmek için bir çok öneri ve teknik bulunmaktadır. Yapılan bu değişiklik genel olarak değerlendirildiğinde TCP protokolünü önemli oranda değiştirmediği değerlendirilmektedir.

İkinci eleştiri ise SYN çerezlerinin TCP eklentilerinin kullanımına izin vermediği yönündedir. Örneğin büyük MSS değerine sahip paketlerin kullanımı gibi. D.J.Bernstein SYN çerezlerinin TCP eklentilerine zarar vermediğini ve zaten bir çok ortamda büyük MSS değerlerinin desteklenmediğini belirtmektedir[3]. Andi Kleen tarafından yapılan SYN çerezleri gerçeklemesi incelediğinde MSS değerinin 8960 sayısına kadar desteklendiği görülmektedir.

MSS (Payload) değeri 8960 olan paketler Jumbo paket olarak adlandırılmaktadır. Yukarıdaki problem aslında büyük MSS değerine sahip paketlerin SYN çerezleri tarafından desteklenmemesinden öte bu paketlerin ağ ortamındaki diğer aktif cihazlar tarafından desteklenmemesidir. Bu açıdan bakıldığında bu durumun SYN çerezleri yöntemi için ciddi bir eksiklik olmadığı düşünülmektedir.

Önemli gördüğümüz diğer bir eleştiri ise SYN çerezlerinin servis kalitesinde ciddi bir düşüşe neden olduğunun söylenmesidir. D.J.Bernstein SYN çerezlerinin servis kalitesini arttırdığını belirtmektedir.

Diğer eleştiriler ise SYN çerezlerinin çok sayıda beklemede kalan bağlantıya neden olduğu ve SYN çerezlerinin beklenmeyen resetlere neden olduğu yönündedir[21].

D.J.Bernstein SYN çerezleri kullanılsa da kullanılmasa da bazı durumlarda bağlantıların beklemede kalabileceğini bu durumun bilgisayar ve ağın aşırı yüklenmesine bağlı olduğunu belirtmektedir. Bağlantının asılı kalması SYN çerezleri yönteminin bir sonucu değildir. Çünkü SYN çerezleri yöntemi TCP'nin ilk el sıkışma yöntemini değiştirmez. İstemci ya da sunucuya ek paket üretme görevi vermez. Sunucular bu durumla geçersiz bağlantıları düşürerek baş ederler.

Andi Kleen tarafından yapılan SYN çerezleri gerçeklemesi SYN listesi dolu olduğunda devreye girmektedir. SYN saldırısının algılanması bu temele dayanır. Bu algılama yöntemi yeteri kadar hassas değildir. SYN saldırısı daha önce algılanmalı ve SYN sırası dolmadan saldırı için SYN çerezleri yöntemi devreye sokulmalıdır.

SYN çerezleri yönteminde; saldırganlar sahte bağlantılar kurabilmek için 32 bitlik çerez değerini tahmin etmelidirler. Başarılı olma ihtimalleri çerez geçerliliğini yitirmeden (zamana bağlı olarak) 4 dakikalık süre içerisinde 4 milyar tahmin seçeneklerinden kaçını test edebilmelerine bağlıdır[13].

Saldırgan SYN çerezleri yöntemi aktif olan bir trafiği inceleyerek parola serisinin bir parçasını elde edebilir. Bu parça içerisindeki ISN değerleri incelenerek artış değerleri hakkında bilgi alabilir ve ACK seli saldırıları düzenleyebilir.

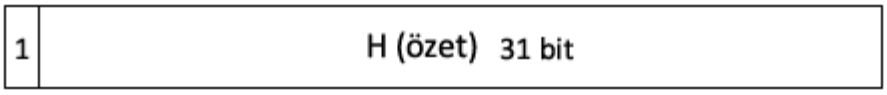
SYN çerezinin önemli ilk 8 biti çıkartıldığında geriye 24 bit kalmaktadır. İlk 8 bit tahmin edilebileceğinden çerezin kırılması 24 bitlik özet değerinin çözülmesine bağlıdır. Bu durumda çakışma olasılığı oldukça yüksektir[22]. Çerez içerisindeki özet değerinin bitsel uzunluğunun 24 bit olması güvenliği azaltmaktadır. Saldırgan 24 bitlik özet değerini çözerek ACK seli saldırıları düzenleyebilir. Çerez içerisindeki özet değerinin bitsel uzunluğunun arttırılması güvenliği arttıracaktır.

5.6 SYN Çerezleri Yöntemini İyileştirmek İçin Önerilen Yöntemler

Bo Hang, Ruimin Hu, Wei Shi[21] D.J.Bernstein tarafından önerilen SYN çerezleri yönteminde oluşan hesaplama karmaşıklığını ve sonuç olarak ek işlemci zamanını azaltmak için aşağıdaki çerez hesaplama yöntemini önermişlerdir.

Bu yöntem diğer bileşenlerinin yanında yeni bir çerez hesaplama algoritması önermektedir. Önerilen yöntem 32 bitlik bir Blowfish şifreleme algoritması kullanmaktadır. Çerezin hesaplanmasında TCP paket başlığı içerisindeki bilgilerin yanında zamanla değişen ve rastgele oluşturulan gizli değerler kullanılmaktadır.

D.J.Bernstein tarafından önerilen 32 bitlik çerez alanında, T (zaman sayacı) ve M (şifreli mss değeri) için 8 bit kullanılır. Geriye kalan 24 bit özet değerini saklar. Özet değerinin bitsel uzunluğunun arttırılması için zaman aşımı değerinin 1 bit olarak belirlenmesi ve kalan bitlerin özet değerine eklenmesi önerilmektedir[21]. Andi Kleen tarafından yapılan SYN çerezleri gerçeklemesinde MSS değeri özet değeri içerisine eklendiğinden bu öneride çerez alanı içerisinde tek başına kullanılmamaktadır.



Şekil 5.11: Önerilen İlk Sıra Numarası Bitsel Şablonu

İlk 8 bitin (MSB) daha kolay tahmin edilebileceği düşünülürse güvenlik 2^{24} ten 2^{31} çıkartılmış olacaktır. Bu yöntemde çerez hesaplanırken kullanılan formüller değiştirilmemiştir. En soldaki bir bit zaman aşımı sertifikası olarak kullanılmaktadır.

D. J. Bernstein tarafından önerilen algoritmaya oranla yeni algoritmada bilgi işleme zamanının ortalamasında %30 azalma söz konusudur. Bir milyon paketin toplam işlem süresi 800ms'den azdır. Sonuç olarak yeni algoritmanın D.J.Bernstein tarafından önerilen algoritmadan daha hızlı olduğu anlaşılmaktadır.

Craig Smith, Ashraf Matrawy SYN çerezleri bitsel şablonunda yer alan zaman damgası (T) için aşağıdaki öneriyi sunmuşlardır. SYN çerezleri yönteminde en kötü olasılık saldırganın çok sayıda makineyi kontrol ederek, zaman unsurunu öngörebilmesi durumudur[23]. Linux uygulamalarında sadece son 4 dakikaya ait çerezler geçerlidir. Bundan hareketle 5 bit olan T (zaman sayacı) 2 bit'e indirgenebilir. Geride kalan zaman bitlerinin çerezin şifrelenmesini güçlendirmek için kullanılması gerekmektedir. Önerilen mevcut yöntem, D.J.Bernstein tarafından önerilen SYN çerezleri yönteminden daha fazla güvenlik sağlamaktadır.

5.7 SYN Çerezleri Yöntemini İyileştirmek İçin Önerdiğimiz Yöntem

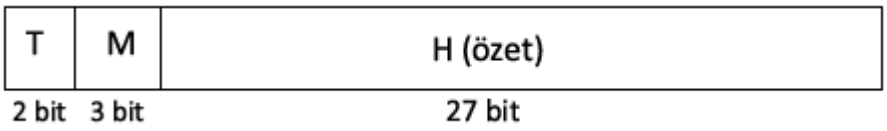
D.J.Bernstein tarafından önerilen SYN çerezleri yöntemi üzerinde 2 değişiklik ön görmekteyiz. Bunlardan birincisi çerez güvenliğinin arttırılması ikincisi ise SYN saldırısının algılanma yönteminin değiştirilmesidir.

5.7.1 Güvenliğinin Arttırılması

Craig Smith, Ashraf Matrawy[23] tarafından SYN çerezleri için önerilen yeni bitsel şablonun, güvenliğin arttırılması için önemli olduğunu değerlendiriyoruz.

SYN çerezleri yönteminde sadece son 4 dakikaya ait çerezler geçerlidir. Bunun sonucu olarak T zaman sayacı için ayrılan 5 bit 2 bite indirilebilir. Elde edilen bu 3 bit, hali hazırda 24 bit olan H değerini güçlendirmek için kullanılabilir. Yeni önerilen yöntemde 32 bitlik SYN çerezleri bitsel şablonunda alt 27 bit, H(özet) değeri olarak kullanılabilecektir. Bu durumda; tahmin edilebilir T (zaman sayacı) ve M (MSS) alanları çıkartıldığında güvenlik 2^{24} ten 2^{27} çıkartılmış olacaktır.

Özet değerinin 24 bitten 27 bite çıkartılması işlemci için ek bir zamana mal olmayacaktır. Andi Kleen tarafından yazılan SYN çerezleri gerçeklemesinde H1 ve H2 özet değerleri hesaplanırken Single Block SHA1 algoritması kullanılmıştır. Bu algoritmanın çıktısı 160 bittir. Sonuç olarak gerçekleme bu bitsel şablondan 27 bitlik bir parça kullanacaktır. Aşağıda önerdiğimiz bitsel şablon görülmektedir.



Şekil 5.12: Önerdiğimiz İlk Sıra Numarası Bitsel Şablonu

Önerdiğimiz bu değişiklik için çekirdek yaması [24] (Craig Smith, Ashraf Matrawy) bulunabilir.

5.7.2 Algılanma Yönteminin Değiştirilmesi

Tez kapsamında Linux çekirdek 2.6.32.69 kullanılmıştır. İlk olarak çekirdek içerisinde TCP katmanının nasıl gerçeklendiğine özet olarak bakılacaktır.

Çekirdek içerisinde TCP katmanı kodlarını içeren temel dosyalar aşağıda listelenmiştir. Bu dosyalara çekirdek içerisinde root/net/ipv4 yolu ile erişilebilir.

tcp.c , tcp_bic.c , tcp_cong.c , tcp_cubic.c , tcp_diag.c , tcp_highspeed.c , tcp_htcp.c , tcp_hybla.c , tcp_illinois.c , tcp_input.c , tcp_ipv4.c , tcp_lp.c , tcp_minisocks.c , tcp_output.c , tcp_probe.c , tcp_scalable.c , tcp_timer.c , tcp_vegas.c , tcp_vegas.h , tcp_veno.c , tcp_westwood.c , tcp_yeah.c

5.7.2.1 Mevcut 2.6.32.69 Çekirdek Gerçeklemesinin İncelenmesi

Sunucuya bir bağlantı talebi gerçekleştiğinde ilk çalıştırılan çekirdek fonksiyonu tcp_v4_conn _request'dir. Bu fonksiyon tcp_ipv4.c dosyası içerisinde bulunmaktadır. Fonksiyon kodları aşağıda verilmiştir.

Çekirdek kodları *https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git /tree/?id=refs/tags/v2.6.32.69* adresinden elde edilmiştir. Satır numaralarının tam olarak eşleşmesi için yukarıda verilen adres kullanılmalıdır.

```
Konum:root/net/ipv4/tcp\_ipv4.c:1213
    int tcp_v4_conn_request(struct sock *sk, struct sk_buff *skb)
    {
        struct inet_request_sock *ireq;
        struct tcp_options_received tmp_opt;
        struct request_sock *req;
        __be32 saddr = ip_hdr(skb)->saddr;
        __be32 daddr = ip_hdr(skb)->daddr;
        __u32 isn = TCP_SKB_CB(skb)->when;
        struct dst_entry *dst = NULL;
    #ifdef CONFIG_SYN_COOKIES
        int want_cookie = 0;
    #else
    #define want_cookie 0
    #endif

        if (skb_rtable(skb)->rt_flags & (RTCF_BROADCAST | RTCF_MULTICAST))
                goto drop;

        if (inet_csk_reqsk_queue_is_full(sk) && !isn) {
    #ifdef CONFIG_SYN_COOKIES
                if (sysctl_tcp_syncookies) {
                        want_cookie = 1;
                } else
```

```
#endif
            goto drop;
    }
    if (sk_acceptq_is_full(sk) && inet_csk_reqsk_queue_young(sk) > 1)
            goto drop;

    req = inet_reqsk_alloc(&tcp_request_sock_ops);
    if (!req)
            goto drop;

#ifdef CONFIG_TCP_MD5SIG
    tcp_rsk(req)->af_specific = &tcp_request_sock_ipv4_ops;
#endif
    tcp_clear_options(&tmp_opt);
    tmp_opt.mss_clamp = 536;
    tmp_opt.user_mss  = tcp_sk(sk)->rx_opt.user_mss;

    tcp_parse_options(skb, &tmp_opt, 0);
    if (want_cookie && !tmp_opt.saw_tstamp)
            tcp_clear_options(&tmp_opt);
    tmp_opt.tstamp_ok = tmp_opt.saw_tstamp;

    tcp_openreq_init(req, &tmp_opt, skb);

    ireq = inet_rsk(req);
    ireq->loc_addr = daddr;
    ireq->rmt_addr = saddr;
    ireq->no_srccheck = inet_sk(sk)->transparent;
    ireq->opt = tcp_v4_save_options(sk, skb);

    if (security_inet_conn_request(sk, skb, req))
            goto drop_and_free;

    if (!want_cookie)
            TCP_ECN_create_request(req, tcp_hdr(skb));

    if (want_cookie) {
#ifdef CONFIG_SYN_COOKIES
            syn_flood_warning(skb);
            req->cookie_ts = tmp_opt.tstamp_ok;
#endif
            isn = cookie_v4_init_sequence(sk, skb, &req->mss);
    } else if (!isn) {
            struct inet_peer *peer = NULL;
            if (tmp_opt.saw_tstamp &&
                tcp_death_row.sysctl_tw_recycle &&
                (dst = inet_csk_route_req(sk, req)) != NULL &&
                (peer = rt_get_peer((struct rtable *)dst)) != NULL &&
                peer->v4daddr == saddr) {
                    if (get_seconds() < peer->tcp_ts_stamp + TCP_PAWS_MSL &&
                        (s32)(peer->tcp_ts - req->ts_recent) >
                                                    TCP_PAWS_WINDOW) {
                            NET_INC_STATS_BH(sock_net(sk),
                            LINUX_MIB_PAWSPASSIVEREJECTED);
                            goto drop_and_release;
```

```
                        }
                }
                else if (!sysctl_tcp_syncookies &&
                         (sysctl_max_syn_backlog - inet_csk_reqsk_queue_len(sk)<
                          (sysctl_max_syn_backlog >> 2)) &&
                         (!peer || !peer->tcp_ts_stamp) &&
                         (!dst || !dst_metric(dst, RTAX_RTT))) {
                        LIMIT_NETDEBUG(KERN_DEBUG "TCP: drop open request
                        from %pI4/%u\n",
                                       &saddr, ntohs(tcp_hdr(skb)->source));
                        goto drop_and_release;
                }

                isn = tcp_v4_init_sequence(skb);
        }
        tcp_rsk(req)->snt_isn = isn;

        if (__tcp_v4_send_synack(sk, req, dst) || want_cookie)
                goto drop_and_free;

        inet_csk_reqsk_queue_hash_add(sk, req, TCP_TIMEOUT_INIT);
        return 0;

drop_and_release:
        dst_release(dst);
drop_and_free:
        reqsk_free(req);
drop:
        return 0;
}
```

Fonksiyon içerisinde SYN çerezleri yöntemine ait birinci kod bloğu aşağıdadır;

```
Konum:root/net/ipv4/tcp_ipv4.c:1222
#ifdef CONFIG_SYN_COOKIES
        int want_cookie = 0;
#else
#define want_cookie 0
#endif
```

Eğer CONFIG_SYN_COOKIES sembolik değişmezi tanımlanmış ise int türünden want_cookie isimli bir değişken tanımlanacak (0 değerine sahip), tanımlanmamış ise want_cookie isimli bir makro oluşturulacaktır. CONFIG_SYN_COOKIES makrosu çekirdek derlenirken config dosyasını oluşturmak için kullanılan make menuconfig komutu ile açılan ve çekirdek parametrelerinin ayarlandığı arayüz de seçili olarak gelmektedir.

Fonksiyon içerisinde SYN çerezleri mekanizmasının devreye girip girmeyeceğinin kararlaştırıldığı (SYN saldırısının algılandığı) blok aşağıda verilmiştir.

```
Konum:root/net/ipv4/tcp_ipv4.c:1236
if (inet_csk_reqsk_queue_is_full(sk) && !isn) {
#ifdef CONFIG_SYN_COOKIES
                if (sysctl_tcp_syncookies) {
                        want_cookie = 1;
                } else
#endif
                goto drop;
        }
```

Sunucuya yapılan her bağlantı isteğinde tcp_v4_conn_request fonksiyonu tekrar çağrılacağından, her çağrımda want_cookie değişkenine 0 değeri atanır. **inet_csk_reqsk_queue _is_full(sk) && !isn** şart ifadesi geriye true(1) değerini döndürürse want_cookie değişkeni 1 değerini alacak diğer durumda want_cookie değişkenin değeri 0 olarak kalacaktır.

want_cookie değişkeninin değeri 1 ise SYN çerezleri mekanizması devreye girecek, 0 ise devreye girmeyecektir. Mevcut çekirdek sürümü want_cookie değişkeninin 1 olduğu durumu SYN seli saldırısı olarak algılamaktadır. Sonuç olarak SYN çerezlerinin devreye girip girmeyeceği her bağlantı isteği için tekrar kontrol edilir.

if ifadesine şart olarak iliştirilen **inet_csk_reqsk_queue_is_full(sk) && !isn** cümleciğinin true değeri üretmesi için **isn** değerinin 0 ve inet_csk_reqsk_queue_is_full(sk) metot çağrım ifadesinin geriye 1 değerini döndürmesi gerekmektedir. Diğer yandan **isn** değeri 0 dan farklı ya da inet_csk_reqsk_queue_is_full(sk) çağrım ifadesi 0 değeri döndürür ise şart cümleciği geriye false(0) değeri döndürür.

inet_csk_reqsk_queue_is_full fonksiyonunun tanımı aşağıda verilmiştir;

```
Konum:root/include/net/inet_connection_sock.h:290
static inline int inet_csk_reqsk_queue_is_full(const struct sock *sk)
{
        return reqsk_queue_is_full(&inet_csk(sk)->icsk_accept_queue);
}
```

Bu fonksiyon sock yapısından (struct) yaratılan temel nesnenin adresini parametre olarak almaktadır. inet_csk fonksiyonu geriye struct inet_connection_sock * türünden bir referans döndürecek ve bu referansın struct request_sock_queue türünden icsk_accept_queue alanı reqsk_queue_is_full metoduna parametre olarak aktarılacaktır.

```
Konum:root/include/net/request_sock.h:228
static inline int reqsk_queue_is_full(const struct request_sock_queue *queue)
{
        return queue->listen_opt->qlen >> queue->listen_opt->max_qlen_log;
}
```

qlen alanı int, max_qlen_log alanı (değişkeni) unsigned char türündendir. Varsayılan olarak max_qlen_log değişkeninin değeri 8 dir. qlen(SYN listesi eleman sayısı) <=255 ise geriye 0 , diğer durumda geriye 1 değeri dönecektir. Redhat temelli dağıtımlarda net.ipv4.tcp_max_syn_backlog değişkenin değeri varsayılan olarak 512'dir.

Her bir bağlantı talebinde tcp_v4_conn_request fonskiyonu içerisinde SYN listesinin dolu olup olmadığı tespit edilmekte eğer bu liste dolu ise SYN çerezleri mekanizması devreye sokulmakta diğer durumda devreye sokulmamaktadır.

Bu yaklaşım, SYN çerezleri gerçeklemesi için en basit yöntemlerden biri olmakla beraber saldırı tespit hassasiyeti azdır. Saldırı altındaki bir sunucu işletim sisteminde SYN sırası hızlı bir şekilde dolar ve SYN sırası dolduktan sonra diğer TCP bağlantıları SYN çerezleri yöntemi kullanılarak gerçekleştirilir. Bu durumda sunucu işletim sistemi hem SYN sırasındaki yarı açık bağlantılar için ACK paketi bekleyecek hem de SYN çerezleri yöntemi ile oluşturulacak bağlantılar için işlemci zamanı harcayacaktır.

5.7.2.2 Önerilen 2.6.32.69 Çekirdek Yapısının Gerçeklenmesi

Orijinal sistemde SYN çerezleri mekanizması SYN listesi dolduğunda devreye girmektedir. Her bir bağlantı isteği için SYN çerezleri mekanizmasının devreye girip girmeyeceği yeniden hesaplanır. Bu yaklaşımda zamana bağlı bir ölçüt kullanılmamaktadır. SYN saldırısının algılanması SYN listesinin dolu olması şartına bağlanmıştır. Test başlığında görüleceği üzere orijinal yaklaşımda saliseler içerisinde SYN listesi dolmakta ve gelen diğer bağlantı isteklerine SYN çerezleri mekanizması ile cevap verilmektedir. SYN listesinin uzatılması bu yüzden SYN seli saldırısına karşı tam bir çözüm olmamaktadır.

SYN seli saldırısının algılanması için sunucu tarafından SYN paketlerine karşılık gönderilen SYN-ACK paketlerinin sayısı ile oluşturulan değişken zaman dilimli bir yöntem önermekteyiz. Önerimiz TCP katmanını oluşturan aşağıdaki dosyalarda değişiklikler içermektedir.

include/net/ request_sock.h, inet_connection_sock.h, tcp.h
net/ipv4/ tcp_input.c, tcp_ipv4.c

tcp_ipv4.c dosyasında bulunan **__tcp_v4_send_synack** fonksiyonu içerisinde sunucu tarafından gönderilen **SYN-ACK** paketleri ve **tcp_input.c** içerisindeki **tcp_rcv_state_ process** fonksiyonu kullanılarak sunucuya gelen **ACK** paketleri sayılacaktır.

Sunucu tarafından SYN paketlerine karşılık gönderilen SYN-ACK paket sayısı (syn_ack değişkeni) inet_csk_reqsk_queue_size(sk)/8 değerine ulaştığında sayma

işlemi bitirilir. inet_csk_reqsk_queue_size() fonksiyonu öneri kapsamında inet_connection_sock.h dosyasına eklenmiştir. Bu fonksiyon SYN listesinin maksimum uzunluğunu döndürür. Test kapsamında SYN listesinin uzunluğu 65536 olarak kullanılacağından sayma işlemi 8192 değerine kadar yapılacaktır.

Sayma işlemi bittiğinde sunucu tarafından gönderilen SYN-ACK paket sayısı ile sunucuya gelen ACK paket sayısı birbirinden çıkartılır. Çıkarma sonucunda elde edilen bu değer 100 değerinden büyük ise bir SYN saldırısı olduğu kabul edilir ve bir sonraki sayım zamanı için SYN çerezleri mekanizması devreye sokulur. Aksi durumda bir sonraki sayma zamanı için SYN çerezleri devrede olmaz. Karşılaştırma işleminde kullanılan 100 değeri farklı saldırılar altında öğrenme süreçlerinden sonra değiştirilebilir.

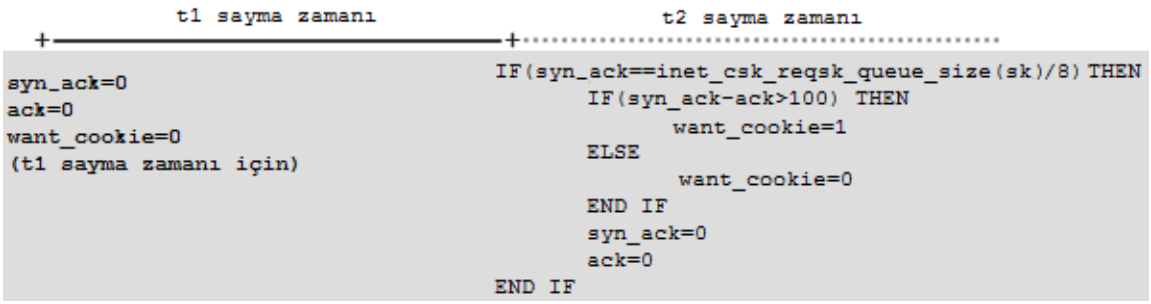
Bir sayım zamanı için SYN çerezleri mekanizmasının aktif ya da pasif olması işlemi orijinal sistemdeki want_cookie değişkeni ile yapılır. want_cookie değişkeni orijinal sistemde yerel bir değişkendir. Bunun nedeni SYN çerezleri mekanizmasının aktif yada pasif olmasına her bir bağlantı için ayrı ayrı karar verilmesidir. Yeni önerdiğimiz sistemde SYN çerezleri mekanizması bir sayım zamanı için aktif yada pasif olacağından want_cookie değişkeni genel bir değişken haline getirilmiştir.

Son aşamada SYN-ACK ve ACK paket sayılarını içeren değişkenler sıfırlanacak ve sayma işlemi tekrarlanacaktır. Her bir sayma işlemi için geçen zaman birbirinden farklı olduğundan değişken zaman dilimleri oluşmuş olacaktır. Bu zaman dilimleri sayım zamanı olarak adlandırılmıştır.

Tanımlanacak değişkenler:

syn_ack: SYN-ACK paket sayısını saklar.
ack: ACK paket sayısını saklar.

Önerilen sistemde temel olarak iki tane değişken tanımı ve kullanımı bulunmaktadır. Şekil 5.13'de önerilen sistemin çalışma modeli görülmektedir.



Şekil 5.13: Yeni Önerilen Sistemin Çalışma Modeli

2.6.32.69 çekirdek sürümünde sırasıyla aşağıdaki değişiklikler yapılmıştır;

1. **ack değişkeninin tanımlanması ve sunucuya gelen ACK paketlerinin sayılması**

   ack değişkeni tcp_ipv4.c ve tcp_input.c dosyalarında kullanılacağından bu değişken tcp.h içerisinde extern anahtarı ile bildirilmiştir.

```
Konum:root/include/net/tcp.h:244(Ekleme)
extern int ack;
```

   tcp_input.c dosyası içerisinde **ack** değişkeni aşağıdaki şekilde tanımlanmıştır.

```
Konum:root/net/ipv4/tcp_input.c:95(Ekleme)
int ack=0;
```

   Sunucu bilgisayara ACK bayraklı bir paket geldiğinde ack değişkeninin değerinin bir arttırılması için tcp_rcv_state_process fonksiyonu kullanılmıştır. Bu fonksiyon bağlantı noktası durumunu güncellemek için diğer çekirdek fonksiyonları tarafından kullanılmaktadır. tcp_rcv_state_process fonksiyonu içerisine aşağıdaki ifade eklenmiştir.

```
Konum:root/net/ipv4/tcp_input.c:5692(Ekleme)
ack++;
```

2. **Yeni SYN Çerezleri Mekanizmasının Kurulması**

   İlk adımda 1222-1226 satırları arasındaki kod bloğu yerleşik olarak global kapsama (tüm fonksiyonların dışına) taşınmış bir sonraki satırda syn_ack değişkeni tanımlanmıştır.

```
Konum:root/net/ipv4/tcp_ipv4.c:88(Ekleme)
  #ifdef CONFIG_SYN_COOKIES
      int want_cookie = 0;
  #else
      #define want_cookie 0
  #endif
  int syn_ack=0;
```

   Sunucu tarafından gönderilen SYN-ACK paketlerinin sayılması için __tcp_v4 _send_synack() fonksiyonu içerisine aşağıdaki ifade eklenmiştir.

```
Konum:root/net/ipv4/tcp_ipv4.c:756(Ekleme)
   syn_ack++;
```

   inet_csk_reqsk_queue_size() ve reqsk_queue_size() fonksiyonları sırasıyla aşağıda belirtilen konumlarda tanımlanmışlardır.

```
Konum:root/include/net/request_sock.h:228(Ekleme)
static inline int reqsk_queue_size(const struct request_sock_queue *queue)
{
        return  1 <<  queue->listen_opt->max_qlen_log;
}
```

```
Konum:root/include/net/inet_connection_sock.h:290(Ekleme)
static inline int inet_csk_reqsk_queue_size(const struct sock *sk)
{
        return reqsk_queue_size(&inet_csk(sk)->icsk_accept_queue);
}
```

tcp_v4_conn_request() fonksiyonu her bir bağlantı talebinde otomatik olarak çağrılmaktadır. Bu fonksiyon içerisine aşağıdaki değişiklikler yapılmıştır.

Aşağıdaki kodlar fonksiyon içerisinden çıkartılmıştır.

```
    Konum:root/net/ipv4/tcp_ipv4.c:1236(Silme)
    if (inet_csk_reqsk_queue_is_full(sk) && !isn) {
    #ifdef CONFIG_SYN_COOKIES
                if (sysctl_tcp_syncookies) {
                        want_cookie = 1;
                } else
    #endif
                goto drop;
        }
```

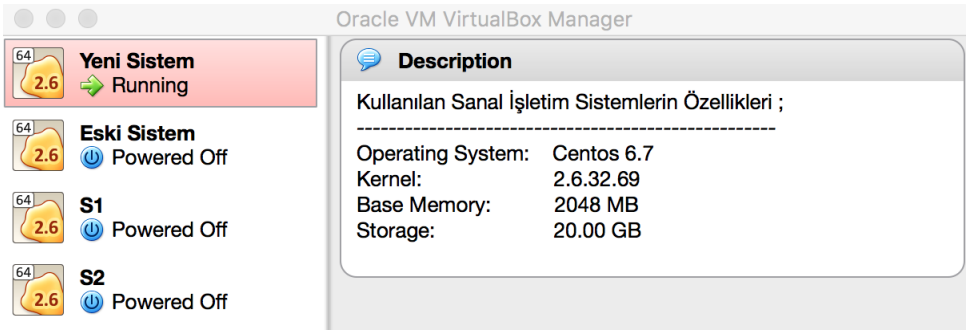
Aşağıdaki kodlar fonksiyon içerisine eklenmiştir.

```
    Konum:root/net/ipv4/tcp_ipv4.c:1236(Ekleme)
     if (syn_ack==inet_csk_reqsk_queue_size(sk)/8){
    if(syn_ack-ack>100)
        want_cookie=1;
    else
        want_cookie=0;
    syn_ack=0;
    ack=0;
}
```

Bölüm 6

Test Ortamı ve Sonuçların Değerlendirilmesi

Testler Oracle Virtual Box 5.0.12 sanallaştırma programı kullanılarak yapılmıştır. Sanallaştırma programının çalıştığı işletim sistemi OSX El Capitan Version 10.11.2 MacBook Pro (2.6 GHz Intel Core i5, 8 GB 1600 MHz DDR3) olarak seçilmiştir. Seçilen Linux dağıtımı Centos 6.7 ve çekirdek sürümü 2.6.32.69 dir. İlgili çekirdek sürümünün seçim nedeni diğer yeni sürümlere göre daha hızlı derlenmesidir. Şekil 6.1'de test aşaması için kurulan sanal işletim sistemleri görülmektedir.



Şekil 6.1: Test Ortamında Kullanılan İşletim Sistemleri

S1 ve S2 işletim sistemleri saldırı yapmak için kullanılmıştır. Tez kapsamında karşılaştırma yapmak için kullanılacak sanal sistemler "Yeni Sistem" ve "Eski Sistem" olarak adlandırılmıştır. Yeni sistem; 2.6.32.69 çekirdeğinin tez kapsamında önerilen yönteme göre düzenlenmiş ve daha sonra derlenmiş bir kopyası ile çalıştırılmaktadır. Eski sistem ise 2.6.32.69 çekirdek sürümünün derlenmiş bir kopyası ile kullanılmaktadır. Sonuç olarak saldırı yapılacak her iki sistem de 2.6.32.69 çekirdek sürümünün sonradan derlenmiş bir kopyasını içermektedir.

Test işlemi iki farklı senaryo altında ayrı ayrı yapılacaktır. Test işlemine geçilmeden önce 2.6.32.69 çekirdek kodlarının nasıl derlendiği ve hangi uygulamaların kurulduğu aşağıda anlatılmıştır.

6.1 2.6.32.69 Çekirdek Sürümünün Derlenmesi ve Gerekli Programların Kurulması

Sırasıyla aşağıdaki komutlar ile çekirdek derleme işlemi gerçekleştirilmiştir.

1.#ls
bin boot dev etc home lib lib64 lost+found media mnt
opt proc root sbin selinux srv sys tmp usr var

2.#cd tmp
3.tmp#wget https://www.kernel.org/pub/linux/kernel/v2.6/longterm/v2.6.32/linux-2.6.32.69.tar.xz
4.tmp# tar -Jxvf linux-2.6.32.69.tar.xz -C /../usr/src
5.tmp# cd /../usr/src/linux-2.6.32.69/

SYN listesi uzunluğunun kalıcı olarak 65536 yapılması için **net/core/request_sock.c** dosyası kapsamında **reqsk_queue_alloc()** fonksiyonu içerisinde lopt->max_qlen_log değişkenine 16 değeri atanmalıdır. Yeni Sistem yapılandırılırken, indirilen çekirdek kodu önerdiğimiz SYN çerezleri yöntemine göre düzenlenmelidir.

yum komutu RHEL (RedHat Enterprise Linux) temelli dağıtımlarda kullanılan paket yönetim aracıdır. Paketleri kurmak, kaldırmak, bağımlıklık problemlerini çözmek ya da bilgi almak için yum aracı kullanılabilir.

6.linux-2.6.32.69# yum install ncurses* gcc* make*
Bu adımda derleme işlemi için gerekli olan ncurses, gcc, make kütüphaneleri kurulmaktadır.

7.linux-2.6.32.69# make menuconfig
Bu komut yeni derlenecek Linux çekirdeğinin özelliklerini ayarlamamıza olanak sağlayan bir arayüz oluşturur bu arayüz ile seçilen özellikler çekirdek yapısına dahil edilirler.

8.linux-2.6.32.69# make
Bu komut ile derleme işlemi gerçekleştirilmektedir.

9.linux-2.6.32.69# make modules
Modül olarak seçili özellikleri derlemek için kullanılır.

10.linux-2.6.32.69# make modules_install install

Sırasıyla modülleri ve çekirdeği masaüstü sisteme yüklemektedir[25]. Sonuç olarak sıkıştırılmış çekirdek imajının /boot klasörü altında oluşturulduğu aşağıdaki komut ile görülebilir.

11.boot#ls -l

```
config-2.6.32.67-504.el6.x86_64
efi
grub
nitramfs-2.6.32.67-504.el6.x86_64.img
initramfs-2.6.32.69.img
lost+found
symvers-2.6.32.67-504.el6.x86_64.gz
System.map -> /boot/System.map-2.6.32.67
System.map-2.6.32.67-504.el6.x86_64
System.map-2.6.32.69
vmlinuz -> /boot/vmlinuz-2.6.32.69
vmlinuz-2.6.32.67-504.el6.x86_64
vmlinuz-2.6.32.69
```

Eski ve Yeni olarak adlandırılmış sunucu işletim sistemleri üzerine PHP, Apache, MySQL servisleri kurulmuştur.

1.# yum install httpd httpd-devel

2.# yum install mysql mysql-server

3.# yum install php-mysql php-pear php-common php-gd php-devel php php-mbstring php-cli

4.Aşağıdaki komutlar lie http, mysqld servisleri başlatılmalıdır.
service httpd start
service mysqld start

5.Bilgisayar her açıldığında ilgili servislerin tekrar başlatılması için aşağıdaki komutlar yazılmalıdır.
/sbin/chkconfig –levels 345 httpd on
/sbin/chkconfig –levels 345 mysqld on

6. TCP 80 bağlantı noktasının açılması için aşağıdaki satırlar çalıştırılmalıdır.
iptables -A INPUT -p udp -m state –state NEW –dport 80 -j ACCEPT
iptables -A INPUT -p tcp -m state –state NEW –dport 80 -j ACCEPT
service iptables save

6.2 Test İşleminin Yapılması ve Sonuçların Değerlendirilmesi

Test işlemi yeni ve eski sanal işletim sistemlerinin saldırı altında kullandıkları CPU ve RAM miktarlarının ölçülmesi ve karşılaştırılmasını amaçlamaktadır. Saldırı 2 farklı yöntem ile yapılmış ve sonuçlar elde edilmiştir.

Saldırı için S1 ve S2 işletim sistemleri üzerinde hping komutu çalıştırılmıştır. Test sürecinde saldırı yapılan sistemlerden bilgi toplamak için **sar** komutu kullanılmıştır. Bu komut işlemci, bellek, diskler, ağ arayüzleri gibi kaynaklardan bilgi toplamak ve raporlamak için kullanılmaktadır.

Bellek kullanım oranlarını elde etmek için **sar -r 1** komutu, işlemci kullanım oranlarını elde etmek için **sar -u 1** komutu kullanılmıştır.

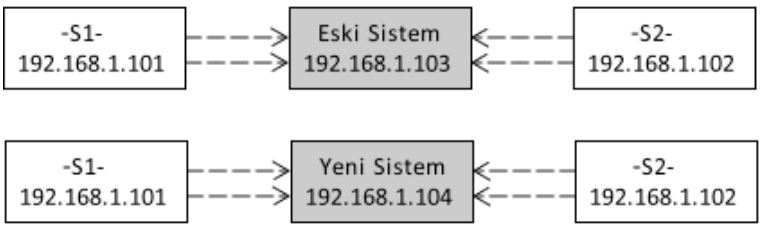
sar komutundan -r parametresi ile edilen çıktıda kbmemfree, kbmemused, %memused, kbbuffers ,kbcached, kbcommit, %commit alanları bulunmaktadır. Test işlemi için bu alanlardan sadece **kbmemused, %memused** alanlarının karşılaştırılması yeterli olacaktır. Bu alanlar bellek kullanım oranlarını sayısal ve yüzde olarak göstermektedir.

sar komutundan -u parametresi ile edilen çıktıda usr, nice, system, sys, iowait, steal, irq, soft, guest, idle alanları bulunmaktadır. Bu alanlardan sadece **% system, % idle** alanlarının karşılaştırılması yeterli olacaktır. system alanı sistem düzeyinde (çekirdek tarafından) kullanılan cpu miktarını, idle alanı işlemicinin boşluk (boşta olma) miktarını gösterir.

6.2.1 Sınırlandırılmış Paket Sayısı İle Saldırı

İşlemciler üzerinde doyum durumu oluşturmadan Eski ve Yeni sistem arasındaki işlemci kullanım oranlarının tespit edilebilmesi için bu test yöntemi kullanılmıştır.

Bu aşamada S1 ve S2 işletim sistemleri kullanılarak eş zamanlı olarak ilk önce Eski daha sonra Yeni sistem'e saldırılar düzenlenmiş ve kullanım oranları elde edilmiştir.



Şekil 6.2: Saldırı Topolojisi

S1 ve S2 sistemleri üzerinde hping komutu aşağıdaki şekilde çalıştırılmıştır;

```
hping3 --rand-source -p 80 -S -i u100 -c 2000000 x.x.x.x
```

-i(interval) seçeneği ile paket gönderimleri arasındaki süre ayarlanabilmektedir. -i u100 ifadesi ile her 100 micro saniyede 1 paket gönderilmesi sağlanmıştır. -c(count) seçeneği ile toplam gönderilecek paket sayısı ayarlanabilir. Bu komut ifadesi ile yaklaşık 2000000 paket hedefe gönderilmektedir. S1 ve S2 sistemleri ile eş zamanlı olarak hem Eski hemde Yeni sisteme yaklaşık toplam 4000000 paket gönderilmiştir. Toplam saldırı süresi 6 dakika olarak ölçülmüştür. Değerlendirme yapmak için saldırı süresi dahil toplam 10 dakika boyunca kullanım oranları kaydedilmiştir.

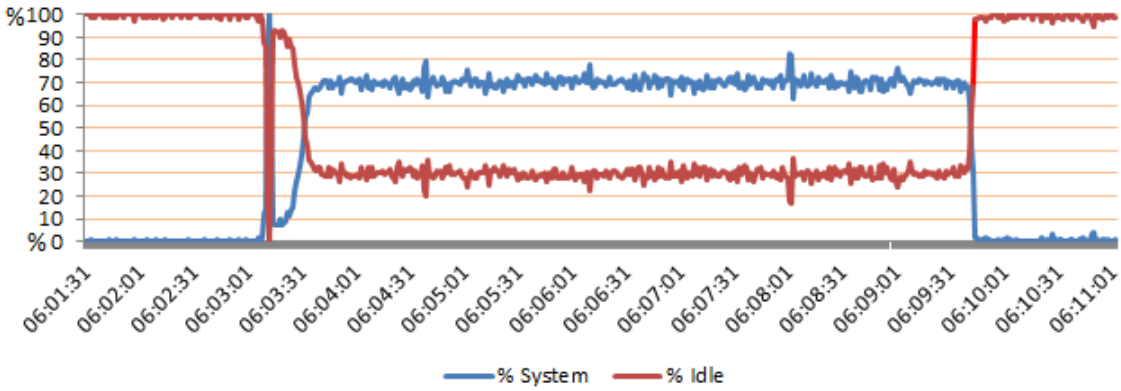
Eski sistemde saldırı öncesi ortalama bellek kullanımı **326897KB (%15,9)**, yeni sistemde bu oran **328125KB (% 15,9)** olarak ölçülmektedir. Saldırı anında Eski sistem ortalama bellek kullanım oranı **563250KB (%27.4)** iken bu oran Yeni sistemde **555819KB (%27)** olarak tespit edilmektedir. Sonuç olarak Yeni sistemin Eski sistemden ortalama **%0.4** oranında daha az bellek kullandığı tespit edilmektedir.

Eski ve Yeni sistemden edle edilen ortalama işlemci kullanım oranları aşağıdaki tabloda görülmektedir.

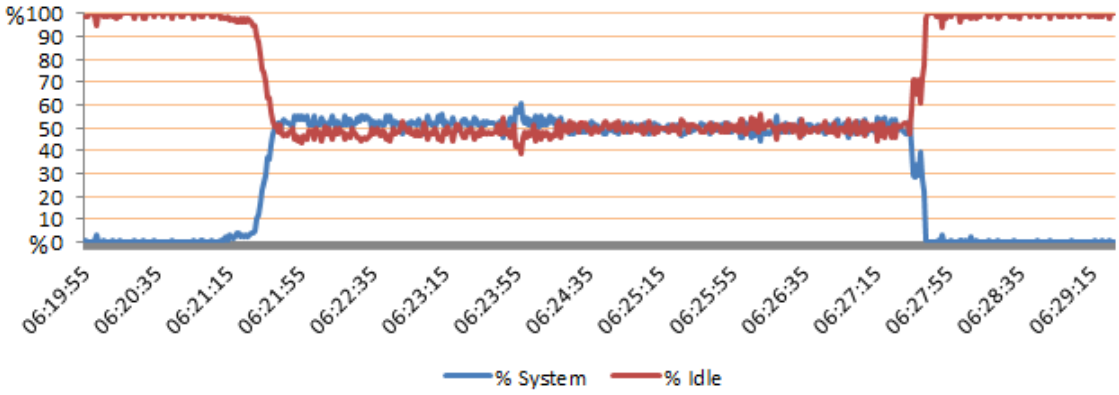
	Saldırı Öncesinde		**Saldırı**		**Saldırı Sonrasında**	
	% system	*% idle*	*% system*	*% idle*	*% system*	*% idle*
Eski Sistem	0,17	99,59	67,21	32,61	0,65	99,09
Yeni Sistem	0,32	99,45	48,35	51,45	0,19	99,52

Şekil 6.3: Ortalama İşlemci Kullanım Oranları

Aşağıda Eski ve Yeni sistemden elde edilen işlemci kullanım oranları grafiksel olarak görülmektedir.



Şekil 6.4: Eski Sistem İşlemci Kullanım Oranları



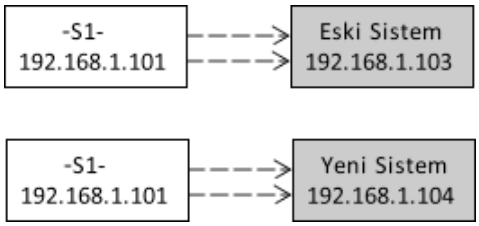
Şekil 6.5: Yeni Sistem İşlemci Kullanım Oranları

Sonuçlar incelendiğinde Eski sistemde saldırı süresince ortalama işlemci boşta olma oranı (% idle) % 32,61 iken Yeni sistemde bu değer % 51,45 olarak ölçülmektedir.

Yeni sistemde saldırı altında değişken zaman aralıklarında SYN listesinin kullanımı SYN çerezleri yöntemi devreye sokularak sınırlandırılmaktadır. SYN listesi çekirdek üzerinde bir bağlı liste türevi olarak gerçeklenmiştir. Eski sistemde saldırı süresince SYN listesi üzerindeki işlemler (nesne yaratma, ekleme, silme, zaman aşımının beklenmesi) yeni sisteme göre oldukça fazladır. Bu durumun iki sistem arasındaki işlemci zamanı farkını oluşturduğu değerlendirilmektedir.

6.2.2 Sınırlandırılmamış Paket Sayısı İle Saldırı

Bu test kapsamında sadece S1 (Tek bir saldırgan) kullanılarak ilk önce Eski daha sonra Yeni sistem'e saldırılar düzenlenmiş ve kullanım oranları elde edilmiştir.



Şekil 6.6: Saldırı Topolojisi

S1 üzerinde hping komutu aşağıdaki şekilde çalıştırılmıştır;

```
hping3 --rand-source -p 80 -S  x.x.x.x --flood
```

Herhangi bir paket sınırlaması yapılmadan hping komutu ile her iki sisteme ayrı ayrı saldırı düzenlenmiştir. Saldırı 10 dakika boyunca sürdürülmüştür. Değerlendirme yapmak için saldırı süresi dahil toplam 13 dakika boyunca kullanım oranları

kaydedilmiştir. –flood seçeneği ile işletim sisteminin mümkün olabilecek en fazla paketi hedef sisteme göndermesi sağlanmaktadır.

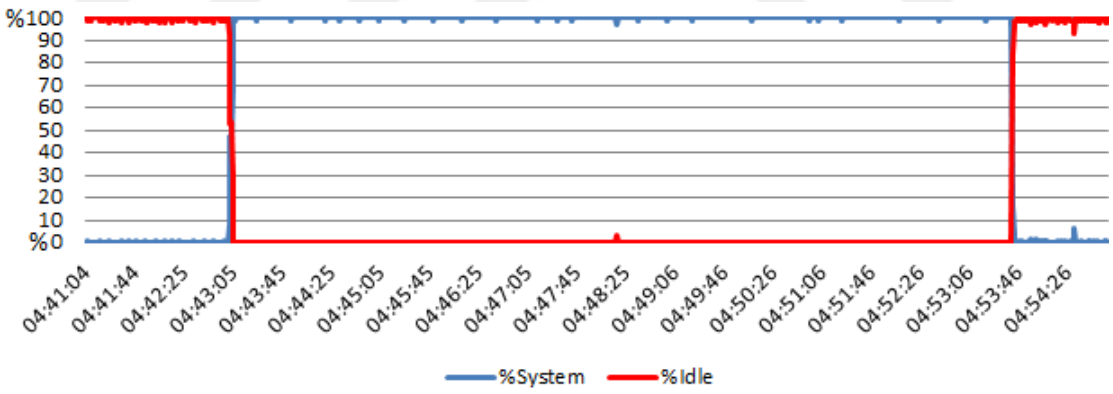
Eski sistemde saldırı öncesi ortalama bellek kullanımı **325777KB (%15,8)**, yeni sistemde bu oran **325766KB (% 15,8)** olarak ölçülmektedir. Saldırı anında Eski sistem ortalama bellek kullanım oranı **577206KB (%28)** iken bu oran Yeni sistemde **563578KB (%27.4)** olarak tespit edilmektedir. Sonuç olarak Yeni sistemin Eski sistemden ortalama **%0.6** oranında daha az bellek kullandığı tespit edilmektedir.

Eski ve Yeni sistemden edle edilen ortalama işlemci kullanım oranları aşağıdaki tabloda görülmektedir.

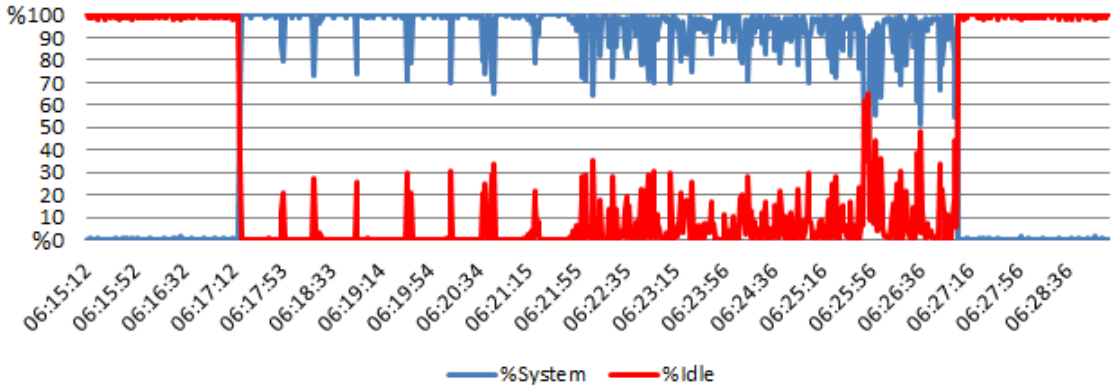
	Saldırı Öncesinde		**Saldırı**		**Saldırı Sonrasında**	
	% system	*% idle*	*% system*	*% idle*	*% system*	*% idle*
Eski Sistem	0,12	99,56	99,59	0,36	0,69	99,07
Yeni Sistem	0,66	99,09	94,74	5,11	0,13	99,65

Şekil 6.7: Ortalama İşlemci Kullanım Oranları

Aşağıda Eski ve Yeni sistemden elde edilen işlemci kullanım oranları grafiksel olarak görülmektedir.



Şekil 6.8: Eski Sistem İşlemci Kullanım Oranları



Şekil 6.9: Yeni Sistem İşlemci Kullanım Oranları

Bu test kapsamında paket sınırlaması yapılmadığı için Eski sistem işlemcisinin doyuma girdiği görülmektedir. Sonuç olarak Yeni sistemin ortalama işlemci kullanım oranının Eski sistemden daha az olduğu görülsede sınırlandırılmış paket sayısı ile yapılan saldırıların iki sistem arasındaki farkı daha doğru bir şekilde ortaya koyduğu düşünülmektedir.

6.2.3 Sonuçlar ve Öneriler

Erişebilirlik Bilgi Güvenliğinin en önemli yapı taşlarından birisidir. Tez kapsamında son yıllarda etkinliği artan ve direk erişebilirliği hedef alan TCP SYN Seli saldırısı ve bu saldırının etkilerini azaltmak için kullanılan SYN çerezleri yöntemi incelenmiştir.

İlk önerimiz SYN çerezleri yönteminin güvenliğinin arttırılması ile ilgilidir. Çerez güvenliği 2^{24} ten 2^{27} çıkartılmıştır.

Hali hazırda Linux çekirdeklerinde kullanılan SYN çerezleri gerçeklemesinin kaynak tüketimini arttıran en önemli eksikliği saldırı tespitinin SYN listesi dolduktan sonra yapılmasıdır.

Mevcut sistemde SYN çerezleri yöntemi bağlantı isteği başına aktif yada pasif yapılmaktadır. Bu durum her bağlantı isteğinde SYN listesinin doluluk oranının test edilmesi ve diğer işlemleri beraberinde getirir. Tez kapsamında önerdiğimiz yeni gerçeklemede, SYN çerezleri yöntemi değişken zaman aralıklarında aktif ya da pasif olarak ayarlanmaktadır.

Tez kapsamında 2 farklı test yöntemi kullanılmıştır. Sınırlı paket sayısı ile yapılan saldırı, işlemcileri doyuma sokmadığı için diğer test yöntemine göre daha kesin sonuçlar üretmektedir. Sonuç olarak sınırlı paket sayısı ile yapılan teste saldırı altında Yeni sistemin Eski sisteme göre daha az işlemci zamanı harcadığı tespit edilmiştir. Bunun yanında saldırı altında bellek kullanım oranları arasındaki fark % 1 altında kalmıştır.

Sonraki çalışmalarda önerdiğimiz değişken zaman dilimlerinin ölçülmesi ve bunun yanında farklı ortamlar için uygun sayma ve karşılaştırma değerlerinin tespit edilmesi yöntemin etkinliğini arttıracaktır.

Kaynakça

[1] Verisign Internet Services Company. *DDoS Saldırıları ve Korunma Yöntemleri.*

[2] Imperva-Cyber Security. *The Top 10 DDoS Attack Trends Discover the Latest DDoS Attacks and Their Implications.*

[3] D.J.Bernstain. Syn cookies.

[4] Wikipedia. *Denial-of-service attack.*

[5] TÜBİTAK Siber Güvenlik Enstitüsü. *TCP/IP Güvenliği Bölüm 1,2.*

[6] B.Demir. *Yazılım Güvenliği.* Kodlab Yayıncılık, İstanbul.

[7] H.Önal. *DNS Hizmetine Yönelik DOS/DDOS Saldırıları.*

[8] Verisign Internet Services Company. *2015 DDOS Saldırı Trendleri.*

[9] K.Burlu. *Bilişimin Karanlık Yüzü.* Nirvana Yayıncılık, İstanbul.

[10] J.Pescatore. *DDoS Attacks Advancing and Enduring: A SANS Survey.*

[11] Dr.R.Çölkesen(Editör:Dr.O. Aliefendioğlu). *Bilgisayar Ağları ve İnternet Mühendisliği.* Papatya Yayıncılık, İstanbul.

[12] H.Tutkun. *Network Sistemleri-Sistem Yöneticisinin El Kitabı.* Seçkin Yayıncılık, İstanbul.

[13] Jon Postel. Syn cookies, an exploration.

[14] G.W.Gordon. Transmission control protocol.

[15] CERT. Tcp syn flooding and ip spoofing attacks.

[16] W. Richard Stevens. *TCP/IP Illustrated, Volume I1994.* Addision Wesley, 1994.

[17] H.Önal. *DOS/DDOS Saldırıları.*

[18] D.J.Bernstain. Syn cookies archive.

[19] Andre Zuquete. Improving the functionality of syn cookies. In *Advanced Communications and Multimedia Security*, pages 57–77. Springer, 2002.

[20] Linux kernel source code.

[21] S.Bellovin. Rfc-1948.

[22] Bo Hang and Ruimin Hu. A novel syn cookie method for tcp layer ddos attack. In *BioMedical Information Engineering, 2009. FBIE 2009. International Conference on Future*, pages 445–448. IEEE, 2009.

[23] Craig Smith and Ashraf Matrawy. Comparison of operating system implementations of syn flood defenses (cookies). In *Communications, 2008 24th Biennial Symposium on*, pages 243–246. IEEE, 2008.

[24] C.Smith. Patch for modifications to linux syn cookies.

[25] N.Koç. *Gömülü Linux Sistemleri*. Pusula Yayıncılık, İstanbul.