Uluslararası Teknolojik Bilimler Dergisi
International Journal of Technological Sciences
Elektronik Dergi, e-ISSN: 1309-1220
Cilt/Volume: 11 Sayı/Number: 2 Sayfa/Page: 118-125 Nisan/April 2019

OpenCL ile FPGA Üzerinde Güvenlik Duvarının Gerçeklenmesi

Mohammed Ridha FAISAL, Tuna GÖKSU*
*Isparta Uygulamalı Bilimler Üniversitesi, Teknoloji Fakültesi, Elektrik-Elektronik Mühendisliği, Isparta

(Alınış Tarihi/Received: 09.01.2018, Kabul Tarihi/Accepted: 10.06.2019)
*İlgili yazar/Corresponding Author: tunagoksu@isparta.edu.tr

Anahtar Kelimeler
OpenCL
FPGA
Güvenlik duvarı

Özet: Bu çalışma, bilgisayar ağlarında kullanılan güvenlik duvarlarının artan veri trafiğini gecikmesiz ve paket kayıpsız işleyebilmelerini sağlamak amacıyla yapılmıştır. Bu amaçla paralel programlamaya imkân veren OpenCL dili kullanılarak FPGA üzerinde paralel mimari ile bir güvenlik duvarı geliştirilmiştir. IPv6 paketleri için kural denetleme işlemleri FPGA üzerinde paralel olarak yapılmıştır. Paket filtreleme işlemi için, ana bilgisayar, IP paketlerini FPGA'e yönlendirmiştir. Kurallar ile paketler arasında eşleşme olup olmadığını belirten bir karar dizisi, FPGA'den ana bilgisayara döndürülerek, IP paketlerinin iletilmesine veya bırakılmasına karar verilmesi sağlanmıştır. Yapılan çalışmada, IPv6 paketleri ile, 100 güvenlik duvarı kuralı için 1 milyon paket/saniyenin üzerinde bir işleme performansı elde edilmiştir. Bu çalışmanın neticesinde geliştirilen üçüncü katman güvenlik duvarının daha üst seviyede güvenlik cihazlarına temel oluşturması hedeflenmiştir.

OpenCL Implementation of Firewall on FPGA

Keywords
OpenCL
FPGA
Firewall

Abstract: This study was conducted to ensure that firewalls used in computer networks can handle increased data traffic without delay and packet loss. For this purpose, a firewall has been developed with parallel architecture on FPGA by using OpenCL language which enables parallel programming. Rule checking for IPv6 packets is done in parallel on FPGA. For packet filtering, the host redirected IP packets to the FPGA. An array of decisions indicating whether there is a match between rules and packets is returned from the FPGA to the host to decide whether to forward or drop IP packets. In this study, we have achieved a processing performance of more than 1 million packets per second for 100 firewall rules with IPv6 packets. The third layer firewall which is developed as a result of this study, is aimed to be a basis for higher level security devices.

1. Giriş

Bilgisayar ve sayısal iletişim teknolojilerindeki hızlı gelişmeler, insanların sosyal yaşamlarında da önemli değişikliklere neden olmuştur. Günümüzde e-devlet, e-ticaret ve sosyal medya uygulamaları kullanımının oldukça yaygınlaşmış olması, kişilerin diğer kişiler, devlet ve özel sektörle olan iletişiminin önemli bir kısmının elektronik ortamda yapılmakta olduğunu göstermektedir. İnternet kullanımı günlük hayatın bir parçası haline gelmiş olup, insanlar iletişime yönelik ihtiyaçlarının büyük bir kısmını İnternet üzerinden karşılamaya başlamıştır. Ancak İnternet ortamında veya bilgisayar ağları üzerinde taşınan verilerin, istenmeyen erişimlere ya da izinsiz kullanımlara karşı korumasız olduğu da bir gerçektir (Check Point, 2018, Cisco, 2018, ENISA, 2018). Ayrıca İnternet veya ağ sistemine bağlı bilgisayar, sunucu vb. cihazlarında her an için bir elektronik saldırıya maruz kalabileceği göz ardı edilemez. Hem taşınan verilerin hem de sistemdeki cihazların saldırılara ve izinsiz işlemlere maruz kalmaması için yapılan çalışmaların geneli, bilgi sistemleri güvenliği ve ağ güvenliği konusudur

(Kizza, 2014). Saldırıların bir kısmının üstesinden gelmek için önemli araçlardan birisi güvenlik duvarıdır (Başkaya, 2010).

Güvenlik duvarı, ağda konumlandırıldığı yer itibariyle, ağ iletişiminde gecikme ve kayıplara sebebiyet vermemek için yüksek işlemci gücüne ihtiyaç duymaktadır. İşlemcilerin daha hızlı işlem yapmasını sağlamak amacı ile çalışma frekansını arttırmak, artan güç tüketimi ve aşırı ısınma problemleri sebebi ile belirli bir noktadan sonra mümkün olamamaktadır. Bu darboğazı aşmak için, geliştiriciler paralel işleme tekniklerine yönelmiş ve neticesinde çok çekirdekli işlemciler geliştirilmiştir. FPGA ise yapısı itibari ile bu özelliği doğal olarak barındırmaktadır (Sarıtaş ve Karataş, 2013).

FPGA veya Grafik İşleme Ünitesi (GPU) ile güvenlik duvarı gerçekleştirilmesine dair değişik çalışmalar mevcuttur (Jedhe vd., 2008, Tuncer, 2010, Gaur vd., 2011, Sahoo vd., 2014, Keskin vd., 2016). FPGA kullanılan ilk çalışmalarda programlamada HDL dilleri tercih edilmiştir. Çalışmalar incelendiğinde, çoğunlukla FPGA veya GPU yanında çevre bileşenlerini daha rahat kullanabilmek amacı ile bir mikroişlemci veya mikro denetleyicinin de kullanıldığı görülmektedir. Bu heterojen yapı, bir işletim sisteminin de kullanılabilmesine imkân sağlamaktadır. Heterojen yapılarda ilk olarak daha çok GPU mevcut olduğundan doğal olarak paralel programlama araçları grafik işlemcilere özeldi. Heterojen sistemler FPGA gibi değişik unsurlar içermeye başlayınca uygulamaların ortam bağımsız hale gelmesi ihtiyacı üzerine OpenCL ortaya çıkmıştır.

Karimi vd. (2013), yaptıkları çalışmalarında, GPU'nun paralel işleme kapasitesini kullanarak, Linux kullanıcı uzayında Iptables'in ilk eşleştirme kuralı mekanizmasını gerçekleştirmişlerdir. Elde ettikleri bulgular, CPU-GPU ile hızlandırılmış kodun, Iptables kodunun CPU sürümüne göre önemli derecede performans artışı sağladığını göstermiştir. Bu çalışma, OpenCL dili ve FPGA kullanılarak yapılması noktası ile önceki çalışmalardan ayrılmaktadır.

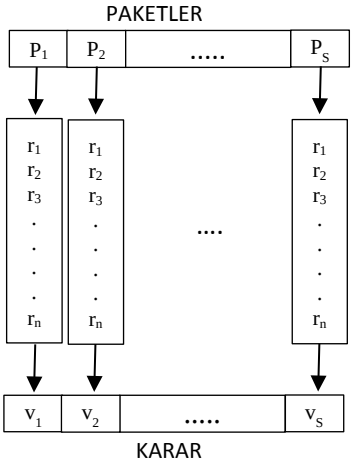
Bu çalışmada, HDL dillerinin kullanımına nispetle, geliştirme süresinin kısaltılması ve geliştirilen yazılımın farklı heterojen sistemler üzerinde çalışabilmesi amacı ile güvenlik duvarının paket filtreleme işlemlerini yapan kısmı OpenCL ile yazılmıştır. Çalışmanın hedefinde bilgisayar ağlarında kullanılan güvenlik duvarlarının artan veri trafiğini gecikmesiz ve paket kayıpsız işleyebilmelerini sağlamak vardır. Geliştirilen sistem FPGA barındıran heterojen bir sistem üzerinde paralel mimari ile oluşturulmuştur ve sonraki çalışmalarda gerçekleştirilebilecek daha üst seviye bir güvenlik duvarına temel oluşturabilmesi amaçlanmıştır. Bu amaçla IPv6 paketleri için kural denetleme işlemleri FPGA üzerinde paralel olarak yapılmıştır. Ana bilgisayar (heterojen sistemdeki CPU) tarafından ağdan alınan IP paketleri, paket filtreleme işlemi için FPGA'e yönlendirilmiştir. Kurallar ile paketler arasında eşleme olup olmadığını belirten bir karar dizisi FPGA'den ana bilgisayara döndürülerek IP paketlerinin iletilmesine veya bırakılmasına karar verilmesi sağlanmıştır.

2. Güvenlik Duvarında Paralel Mimariler

Gün geçtikçe artan veri trafiği, güvenlik duvarlarının paket/saniye ile ölçülen veri (paket) süzme hızlarının artmasını gerekli kılmıştır. Güvenlik duvarlarının paralel programlanması bu amaca ulaşmak için uygun bir yöntemdir. Güvenlik duvarı iki şekilde paralelleştirilebilir: veri-paralel ve işlev-paralel. Bu iki yöntemin birlikte kullanımı ile de Hibrit model isminde üçüncü bir yöntem oluşturulmuştur.

2.1. Veri-Paralel Güvenlik Duvarı

Paralel programlama terminolojisine göre, verileri (paketleri) güvenlik duvarı düğümleri arasında dağıtan bir tasarım veri paralel olarak kabul edilir. Veri-paralel güvenlik duvarı sistemi, paralel bağlanmış birden fazla özdeş güvenlik duvarından oluşturulur. Her bir güvenlik duvarı düğümü, diğerlerinden bağımsız bir şekilde çalışmasını sağlayan, yerel kural setine ihtiyaç duyar. Bu yerel kural setleri, genel güvenlik ilkesinin eksiksiz bir kopyası olmak zorundadır. Gelen paketler daha sonra güvenlik duvarı düğümleri arasında dağıtılır, böylece bir güvenlik duvarı düğümü belirli bir paketi işler. Bu sayede farklı paketler paralel olarak işlenir ve tüm paketler tüm güvenlik politikasıyla karşılaştırılmış olur (Şekil 1.) (Fulp, 2006).

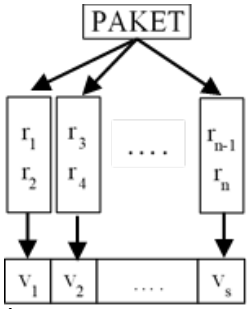


Şekil 1. Veri-Paralel Güvenlik Duvarı

2.2. İşlev-Paralel Güvenlik Duvarı

İşlev-paralel güvenlik duvarı tasarımı güvenlik duvarı düğümleri dizisinden oluşur. Bu tasarımda, filtrelenmesi istenilen trafik, tüm güvenlik duvarı düğümlerine tekrarlanır. Güvenlik duvarı düğümleri tarafından paketler işlendikten sonra, her bir düğümün denetleme sonucu CPU üzerinde çalışan ana bilgisayar programına gönderilir (Şekil 2.). Bir paket üzerinde birden fazla karara sebep verilmemesi amacıyla, güvenlik duvarı düğümlerinin neticeleri sıralı olarak işlenir (Fulp, 2006).

Bir paket aynı anda birden fazla iş parçacığı tarafından işlenir. Her bir iş parçacığı (güvenlik duvarı düğümü) kural setinin bir bölümünü paket üzerinde kontrol eder. Paketler ana bilgisayar programı tarafından OpenCL cihazının üniversal belleğine kopyalanır. Kural veritabanın düğümü ilgilendiren kısmı, OpenCL cihazının sabit belleğinde saklanır. Tüm iş parçacıkları yürütmeyi tamamladıktan sonra oluşan karar dizisi cihazdan ana bilgisayar programına kopyalanır (Reddy vd, 2013).

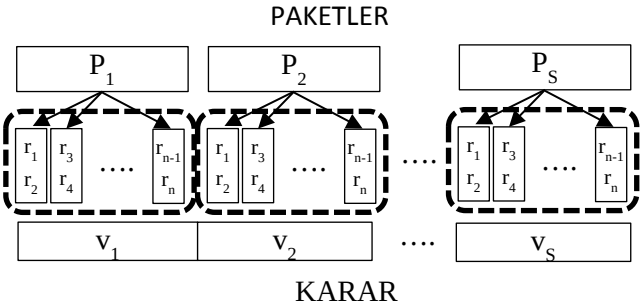


Şekil 2. İşlev-Paralel Güvenlik Duvarı

2.3. Hibrit Model

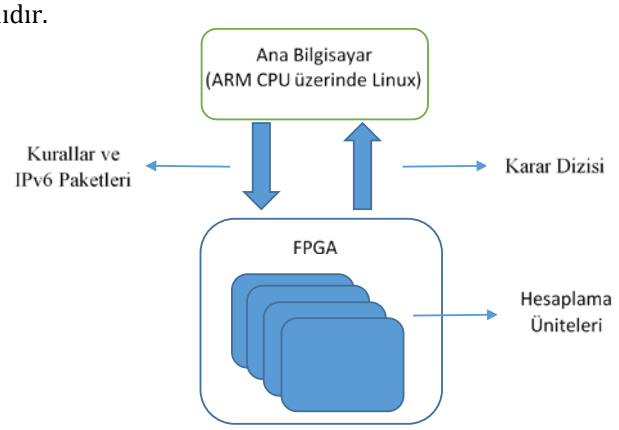
Veri-paralel ve işlev-paralel tasarımlar, FPGA mimarisinin hesaplama yetenekleri ile tam olarak eşleşmemektedir. Veri-paralel tasarım, güvenlik duvarında saniyede işlenebilen paket sayısını artırır. Buna karşılık işlev-paralel tasarım ise paket işleme gecikmesini azaltır. FPGA'ın yüksek paralel hesaplama yetenekleri kullanılarak hem saniyede işlenebilen paket sayısı arttırılabilir hem de paket işleme gecikmesi azaltılabilir. Böylece daha iyi bir model elde edilir (Şekil 3.) (Reddy vd, 2013).

Hibrit modelde bir paket kümesi biriktirilir ve her paket FPGA üzerinde farklı bir blokla paralel olarak işlenir. Bir blok içinde, aynı anda bir paket birden çok iş parçacığı tarafından işlenir ve her iş parçacığı kural setinin bir bölümünü kontrol eder. Kural veri tabanı, OpenCL cihazının üniversal belleğinde tutulur. Sonuçta tüm iş parçacıkları yürütmeyi tamamladıktan sonra paketlerin ne olacağını belirleyen karar dizileri cihazdan ana bilgisayar programına kopyalanır (Reddy vd, 2013).



Şekil 3. Hibrit Model

Hibrit model kullanılırken paketler ile kurallar arasında eşleşmenin varlığının yanında birden fazla eşleşme durumunda hangi paketin hangi kural ile eşleştiği ve eşleşme sırası da önemli ise kodlamada dikkatli olunmalı ve gerekli uyarlamalar yapılmalıdır.



Şekil 4. Güvenlik Duvarı Blok Diyagramı

3. OpenCL Cihazı Çekirdek Kodları

Bu çalışmada ikinci bölümde bahsi geçen modellerden ilki olan veri-paralel güvenlik duvarı gerçekleştirilmiştir. Hibrit model, OpenCL'de atomik işlemler olarak adlandırılan operasyonları gerektirmekte ve sonraki çalışmalarda hayata geçirilmesi planlanmaktadır. Geliştirilen güvenlik duvarına ait blok diyagram Şekil 4.'te verilmiştir. Altera DE1-SoC FPGA kartı üzerinde bulunan ARM tabanlı (Cortex-A9) işlemci üzerinde çalışan Linux işletim sistemi, IPv6 paketlerini ve kural dizisini FPGA'e (Cyclone V SoC 5CSEMA5F31C6) göndermektedir. FPGA üzerinde bulunan, OpenCL ile oluşturulmuş, hesaplama üniteleri, paketler ile kuralları karşılaştırmakta ve eşlemeye dair neticeleri işletim sistemine geri döndürmektedir. Paketler ve kuralların eşleşmesinde kontrol edilenler, temel üçüncü katman bir güvenlik duvarının yaptığı paket filtreleme işleminde olduğu şekilde, kaynak ve hedef IPv6 adresleri, kaynak ve hedef bağlantı (port) numaraları ile protokol numarasıdır.

Altera DE1-SoC FPGA kartı için geliştirilmiş Linux dağıtımında IPTables desteği eklenmemiştir. Geliştirilen uygulamada IPTables tarafından kullanıcı uzayındaki kuyruğa eklenecek olan ağ trafiği suni olarak rastgele verilerden üretilmiştir. Ağ trafiğinin FPGA üzerindeki güvenlik duvarlarına gönderilerek paralel olarak işlenmesini temsil etmek amacıyla, kaynak IP adresi, hedef IP adresi, kaynak bağlantı noktası numarası ve hedef bağlantı noktası numarasını sakladığını varsaydığımız değişkenler tanımlanmıştır. Bu değişkenlerdeki her bir elemanın boyutu saklayacağı veri türüne uyumlu olmalıdır. IPv4 adresleri 32 bit, IPv6 adresleri ise 128 bit uzunluğundadır. OpenCL tamsayı veri tiplerinin en genişi 64 bit uzunluğundadır. Dolayısı ile IPv6 adresleri iki elemanlı diziler içerisine saklanmak zorundadır. Bağlantı noktası numaraları 16 bit uzunluğunda ve protokol bilgisi de 8 bit uzunluğundadır. Belirtilen bilgileri grup halinde ana bilgisayardan FPGA'e transfer edebilmek amacıyla Şekil 5'te gösterilen yapı tanımlanmıştır.

Bu yapıda dummy isminde fazladan değişken tanımlanmıştır. Bu tanımlamanın sebebi, "Intel® FPGA SDK for OpenCL™ Best Practices Guide" kılavuzunda belirtilen, yapı hizalamasının gerçekleştirilmesidir (Intel, 2017). Bu dokümana göre, yapı boylarının byte cinsinden tüm yapı elemanlarının en küçük ortak katı olarak ayarlanması performansı olumlu etkilemektedir. Oluşturulan yapının normal boyutu 38 byte'dır. Yapının boyutunu, tüm elemanları en küçük ortak katı olan 40 byte olacak şekilde ayarlamak için iki byte uzunluğunda kullanılmayan bir eleman eklenmiştir.

```
1.  typedef struct{
2.    ulong sourceIP[2];
3.    ulong destinationIP[2];
4.    ushort sourceport;
5.    ushort destinationport;
6.    ushort protocoltype;
7.    ushort dummy;
8.  }Package;
```

Şekil 5. IPv6 paketleri için kullanılan yapı

4. Ana Bilgisayar OpenCL Kodları

OpenCL programlarında ilk olarak programlanacak platformun varlığı kontrol edilmelidir. Bulunan platform sorgulanarak barındırdığı OpenCL cihazlarının sayısı öğrenilir. GPU'lardan farklı olarak FPGA için çekirdek kodu çalışma anında derlenemez. Bu sebeple önceden Altera OpenCL derleyicisi AOCL ile çekirdek kodu derlenmiş olmalıdır. Sonraki aşamada derlenmiş çekirdek kodu ile FPGA konfigüre edilir. Bu işlemleri takiben sırasıyla içeriğin, komut kuyruğunun ve çekirdeğin oluşturulması işlemleri gerçekleştirilir. Programımızda bir anda işlenecek paket sayısının parametrik olması amacı ile paket sayısını gösteren N isimli bir değişken tanımlanmıştır. Platformda bulunan OpenCL cihazlarının paketleri eşit sayıda bölüşmeleri için gerekli kod yazılmıştır. Elimizdeki geliştirme kartı sadece bir tane OpenCL cihazından oluştuğu için bütün paketler bu tek cihazda işlenmektedir. Yazılan kod bu anlamda geliştirme kartından bağımsız olup birden fazla cihaz bulunması durumunu desteklemektedir.

OpenCL cihazına paketlerin ve güvenlik duvarı kurallarının gönderilebilmesi ve karar dizisinin de okunabilmesi amacıyla gerekli tampon bellekler oluşturulmuştur. Gönderme tampon bellekleri rastgele paketlerle doldurularak OpenCL cihazına gönderilmiştir. Burada her bir çekirdeğin bir paketi kural sayısı kadar kural için kontrol etmesi tasarlandığı için, kural sayısının çekirdekler tarafından bilinmesi gerekmektedir. Bu amaçla paketler ve kuralların ardından kural sayısı da OpenCL cihazına transfer edilmiştir. Gerekli verinin cihaza gönderilmesinin ardından çekirdeklerin çalışması sağlanmış ve netice (karar dizisi) cihazdan okunmuştur.

İşlemlerin doğruluğunu test etmek amacıyla karşılaştırma işlemi SoC FPGA üzerindeki ARM işlemciye de yaptırılmıştır. İşlemcinin çalışma süresi, verilerin cihaza transfer süresi ve çekirdeklerin çalışma süresi OpenCL ve Linux tarafından sağlanan arayüz fonksiyonları (getStartEndTime()ve getCurrentTimestamp()) tarafından ölçülerek kaydedilmiştir.

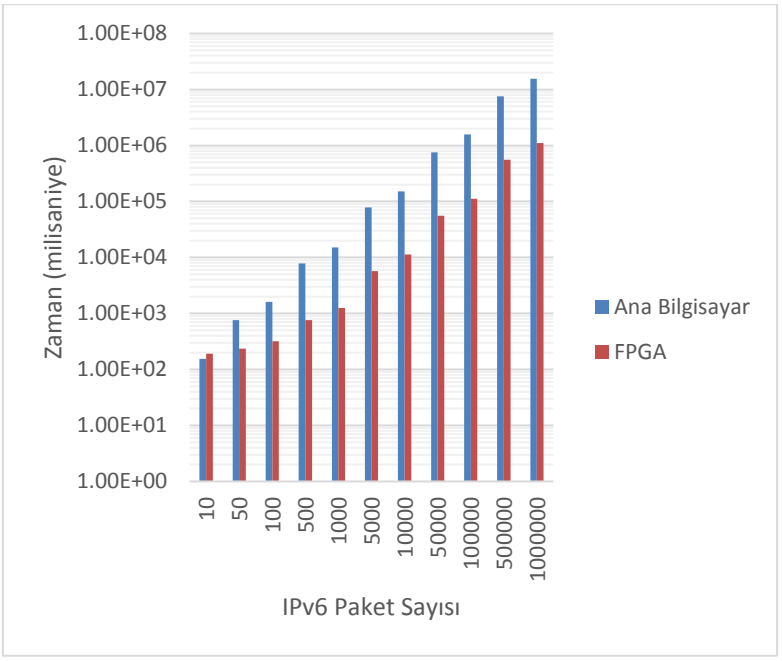
5. Araştırma Bulguları

Bu çalışma ile bilgisayar ağlarında kullanılan güvenlik duvarlarının artan veri trafiğini gecikmesiz ve paket kayıpsız işleyebilmelerini sağlamak amacıyla yeni bir yazılım geliştirilmiştir. Bu amaçla paralel programlamaya imkân veren OpenCL (sürüm 1.2) dili ile Altera DE-1 SoC Kartı (Cyclone V SoC 5CSEMA5F31C6 FPGA) üzerinde paralel mimari ile bir güvenlik duvarı tasarlanmıştır.

Altera SoC FPGA için geliştirilen çekirdek Linux işletim sistemi üzerinde çalışan AOCL (sürüm 17.0.1) derleyici tarafından derlenmiş ve geliştirme kartına aktarılmıştır. Derleyicinin ürettiği tahmini kaynak kullanımı raporuna göre kaynak kullanımları, ALUT (Adaptive Look-Up Table, Uyarlanabilir Arama Tablosu) için %52 ve FF (flip-flop) için %35 olarak görülmektedir.

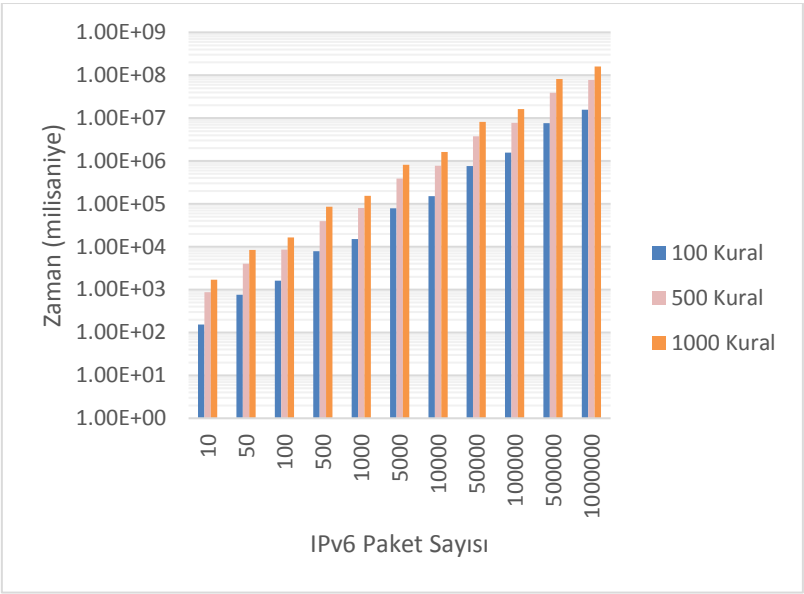
Uygulamanın performansının test edilebilmesi amacı ile farklı sayıdaki kurallar için FPGA ve ana bilgisayar üzerindeki hesaplama süreleri ölçülmüştür. Bunun yanı sıra IP paketlerin ana bilgisayardan FPGA'e aktarım süresi de gözlenmiştir. Bu sürelerin ölçümünde IP paketleri için versiyon 6 kullanılmıştır. Olası en kötü durumu analiz etmek için kurallar ile paketlerin eşleşmediği durumlar dikkate alınmıştır. Hesaplama süreleri kullanılarak ana bilgisayar ve FPGA için hesaplama sürelerine ait grafikler oluşturulmuştur. Birbirinden çok uzak paket sayıları seçildiği ve çalışma süreleri de bu sebeple birbirinden uzak neticelendiği için tüm grafiklerde zaman ekseni logaritmik olarak seçilmiştir.

Şekil 6'da, farklı sayıda paketlerin 100 güvenlik duvarı kuralı için, SoC FPGA üzerinde bulunan ARM tabanlı işlemci tarafından tek çekirdek üzerinde işlenme süreleri ile FPGA üzerinde işlenme süreleri kıyaslanmıştır. Bu grafiğe göre 10 paket kadar bir uzunluk için ana bilgisayar daha hızlı işlem gerçekleştirmektedir. 10'dan daha büyük paket sayıları için FPGA daha hızlı işlem yapmaktadır.

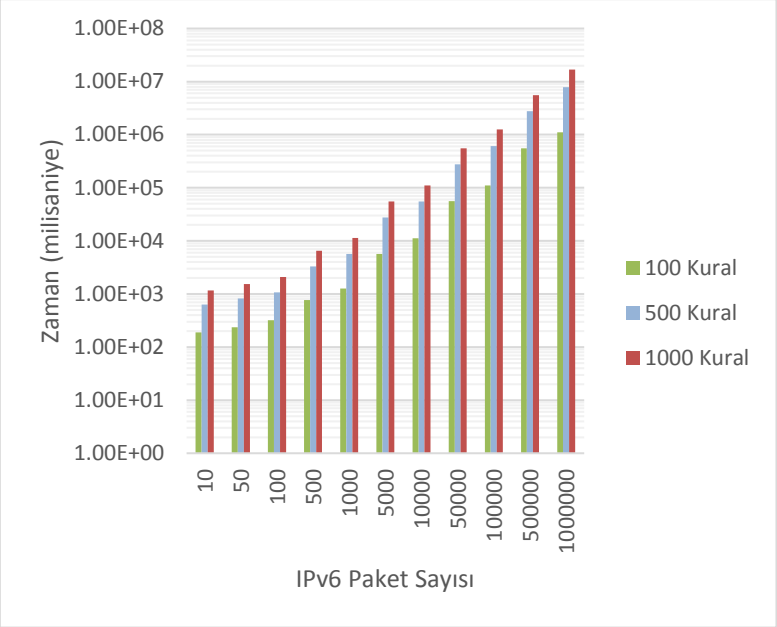


Şekil 6. Ana bilgisayarda ve FPGA'de 100 kural için paketlerin işlenme süreleri

Paket sayıları arttıkça işlem süresi ana bilgisayar için doğrusal olarak artmaktadır. Bu durum işlemcinin yapısı gereği beklenen doğal neticedir. FPGA için işlem sürelerindeki artış 100 pakete kadar doğrusal değildir. 1 milyon paket için süreler incelendiğinde FPGA 1 saniyeden biraz uzun bir sürede işlemini tamamlarken ARM tabanlı işlemci 15,5 saniyeden daha uzun bir sürede işlemi gerçekleştirmiştir. Elde edilen verilerden, 500 paketten sonraki doğrusal kısım kullanılarak, ana bilgisayarın yaklaşık 15,1 µs/paket hızında, FPGA'in ise yaklaşık 0,8 µs/paket hızında işlem yaptığı görülmektedir. Gerçekleştirilen sistemin 100 kural için 1 Mpps değerine ulaştığını söylemek mümkündür. Şekil 6'da FPGA üzerinde paketlerin işlenme sürelerinde 100 pakete kadar yüksek bir değişiklik gözlenmemektedir. Bu durum FPGA içerisinde oluşturulan paralel işleme sayesinde mümkün olmaktadır. Son olarak Şekil 7 ve Şekil 8'de ana bilgisayar ile FPGA için farklı kural sayılarına göre hesaplama süreleri kendi içerisinde kıyaslanmıştır.



Şekil 7. Ana bilgisayar için farklı kural sayılarında işleme süreleri



Şekil 8. OpenCL cihazı için farklı kural sayılarında işleme süreleri

6. Sonuçlar ve Tartışma

Yapılan çalışmada elde edilen sonuçlar 100 kural için 1 milyon paket/saniye paket filtreleme performansının elde edildiğini göstermektedir. Karimi vd. (2013), çalışmalarında, 10000 kural için GPU'daki paket filtrelemesinin genel performansı 400.000 paket/saniye civarındadır. Elde edilen performans bu çalışma ile kıyaslanacak olursa düşük kalmış gibi görülebilir fakat Karimi vd. (2013) IP paketlerinin versiyonundan bahsetmemişlerdir. IPv4 ile IPv6 adresleri arasında ciddi bir uzunluk farkı vardır. IPv6 adresleri IPv4 adres uzunluğunun 4 katıdır. Bu uzunluk farkı performansı ciddi şekilde etkileyen bir unsurdur. Bu çalışmada daha önce de belirtildiği üzere IPv6 paketleri kullanılmıştır. Yine önceki çalışmalarla (Karimi vd., 2013, Widianto vd., 2015) benzer olarak, çok düşük sayıdaki paketler için CPU paket filtreleme performansı daha iyidir. Bunun sebebi, paketlerin ana bilgisayardan OpenCL cihazına aktarılmasının kayda değer bir süre almasıdır.

Bu neticeler ışığında yapılan çalışmada yaklaşık 1000 paket ve daha fazlası için FPGA ile IP paketlerin güvenlik duvarı paket filtrelemesi işlemlerinin, ana bilgisayar ile işlemeye göre daha avantajlı olduğu ortaya çıkmıştır. Daha az sayıdaki paket için, verilerin ana bilgisayardan FPGA'e transfer süresinin uzunluğundan dolayı, verileri ana bilgisayar üzerindeki çekirdeklerde paralel olarak işlemek daha avantajlı görülmektedir. Buradaki sayılar ve süreler elimizdeki mevcut Altera DE-1 SoC geliştirme kartına göre verilmiş olup, farklı model FPGA'lar için paket sayıları farklılık gösterecektir. Genel sonuç itibariyle FPGA ile güvenlik duvarının paralelleştirilmesi performansta artış sağlamıştır. Bu çalışmanın, durum denetlemeli güvenlik duvarı ve daha üst katmandaki işlemler için tatbik edilerek ilerletilmesi düşünülmektedir.

7. Kaynaklar

Ahmed, T. (2011). OpenCL framework for a CPU, GPU, and FPGA Platform, Yüksek Lisans Tezi, University of Toronto, 85s, Toronto, Kanada.

Ajami, R., ve Dinh, A. (2011). Design a hardware network firewall on FPGA. Canadian Conference on Electrical and Computer Engineering (CCECE) 674-678. Niagara Falls, Kanada.

Anshuman, V., Helal, A. E., Krommydas, K., & Feng, W. C. (2016). Accelerating Workloads on FPGAs via OpenCL: A Case Study with OpenDwarfs. Computer Science Technical Reports.

Banger, R. ve K. Bhattacharyya (2013). OpenCL programming by example, Birmingham, Packt Publishing Ltd.

Başkaya, O. (2010). OSI katman modelinde 3. katman güvenlik duvarı uygulaması. Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, 95s, Ankara.

Beal, V. (2010). The Differences and Features of Hardware and Software Firewalls. https://www.webopedia.com/DidYouKnow/Hardware_Software/firewall_types.asp. (2017.10.10)

Belkhede, A., Sonawane, S., Khirsagar, C., ve Sapkal, D. D. (2016). Firewall Engine Based On Graphics Processing Unit. International Journal Of Engineering Sciences & Research Technology, 5,3,424-429

Carriero, N., ve Gelernter, D. (1989). How to write parallel programs: A guide to the perplexed. ACM Computing Surveys (CSUR), 21,3, 323-357.

Chek Point, 2018. 2018 Security Report, 47s.

Cisco, 2018. Annual Cybersecurity Report, 68s.

Çelik Ahmet (2010). Paralel Bilgisayar Sistemlerinin Performans Analizi. Yüksek Lisans Tezi, Dumlupınar Üniversitesi, Fen Bilimleri Enstitüsü, 156s, Kütahya.

ENISA, 2018. Threat Landscape Report, 139s.

Ercan, U., Akar, H., ve Koçer, A. (2013). Paralel programlamada kullanılan temel algoritmalar. Akademik Bilişim'13, 23-25, Malatya.

Eskikaya B. (2012). Distributed OpenCL-OpenCL Platformunun Ağ Ölçeğinde Dağıtılması. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, 71s, İstanbul.

Fulp, E. W. (2006). Parallel firewall designs for high-speed networks. 25TH IEEE International Conference on Computer Communications, 1-4, Barcelona, İspanya.

Gaster, B., L. Howes, D. R. Kaeli, P. Mistry ve D. Schaa (2012). Heterogeneous Computing with OpenCL: Revised OpenCL 1.2 Edition, Massachusetts, Morgan Kaufmann.

Gaur, M. S., Laxmi, V., Cahndra, K., ve Zwolinski, M. (2011). Acceleration of packet filtering using GPGPU. 4th international conference on Security of information and networks, 227-230, Sidney, Avusturalya.

Intel, (2017) FPGA SDK for OpenCL, Best Practices Guide.

Jedhe, G. S., Ramamoorthy, A., ve Varghese, K. (2008). A scalable high throughput firewall in FPGA. 16th International Symposium on Field-Programmable Custom Computing Machines, 43-52. Palo Alto, Amerika.

Karimi, K., Ahmadi, A., Ahmadi, M., ve Bahrambeigy, B. (2013). Parallell Implementation of Linux Packet Filtering. The CSI Journal on Computer Science and Engineering, 11,2, 24-30.

Keskin, S., Erdogan, H. T., ve Kocak, T. (2016). Graphics processing unit based next generation DDoS prevention system. 4th International Symposium on Digital Forensic and Security (ISDFS), 59-62, Little Rock, Amerika.

Kizza, J. M. (2014). Computer network security and cyber ethics, North Carolina, McFarland & Company.
Reddy, K. C., Tharwani, A., ve Krishna, C. (2013). Parallel firewalls on general-purpose graphics processing units. arXiv preprint arXiv:1312.4188.

Sahoo, A. K., Das, A., ve Tiwary, M. (2014). Firewall engine based on graphics processing unit. IEEE International Conference on Advanced Communications, Control and Computing Technologies, 758-763, Tamilnadu, Hindistan.

Sarıtaş, E., ve Karataş, S. (2013). Her yönüyle FPGA ve VHDL, Ankara, Palme Yayıncılık.

Tuncer, T. (2010). Bilgisayar Ağları İçin Saldırı Tespit Sistemi Tasarımları ve FPGA Ortamında Gerçekleştirilmesi. Doktora Tezi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elazığ.

Widianto, A. R., Lim, C., ve Kho, I. E. (2015). Improving performance of intrusion detection system using OpenCl based general-purpose computing on graphic processing unit (GPGPU). 3rd International Conference on New Media (CONMEDIA), 1-5, Banten, Endonezya.