KTÜ Bilgisayar Mühendisliği
Siber Güvenlik Laboratuvarı
Nesnelerin İnternetinde Yönlendirme Protokollerine Yönelik Güvenlik Analizi

1.Giriş

Nesnelerin İnterneti (IoT) teknolojileri, düşük güçlü cihazların birbirleriyle ve merkezi sunucularla haberleşmesini sağlayarak birçok alanda yaygın kullanılmaktadır. Ancak bu cihazların sınırlı donanım kaynaklarına sahip olması, onları güvenlik saldırılarına karşı oldukça hassas hale getirmektedir. Özellikle yönlendirme protokolleri (Routing Protocols), ağın sağlıklı çalışması için kritik rol oynadığından, bu protokollere yapılan saldırılar ağın bütün performansını olumsuz etkileyebilir.

Bu deneyde öğrenciler, Contiki işletim sistemi ve Cooja simülatörü kullanarak küçük ölçekli bir IoT ağı oluşturacak, bu ağda kullanılan RPL (Routing Protocol for Low-Power and Lossy Networks) protokolünün güvenlik açısından davranışlarını inceleyeceklerdir. Ağ, bir sunucu (sink node), on istemci (client node) ve bir saldırgan düğümden oluşacaktır. Saldırgan düğüm, normal protokol akışını bozarak ağın performansını olumsuz etkilemeye çalışacaktır.

Çalışmada iki saldırı senaryosu ele alınacaktır:

- Flooding (Taşkın Trafik) Saldırısı: Saldırgan düğüm, olağandışı yüksek hızda ve yoğunlukta paket göndererek ağı gereksiz trafiğe boğar. Bu saldırı sonucunda bant genişliği tükenir, kuyruklar dolar, normal istemcilerin paketleri kaybolur ve enerji tüketimi artar. Ağda gecikme (latency) yükselir ve güvenilirlik (PDR) düşer.

- Blackhole (Kara Delik) Saldırısı: Saldırgan düğüm, yönlendirme ağacında kritik bir konumda bulunur ve belirli bir süre normal çalıştıktan sonra iletmesi gereken paketleri durdurur. Bu şekilde, saldırganın üzerinden geçen trafiğin tamamı “yutulur” ve hedef sunucuya ulaşamaz. Bu saldırı, özellikle tek çıkış noktası olan alt ağlarda ciddi paket kayıplarına neden olur.

Öğrenme Çıktıları

Bu laboratuvar çalışmasının sonunda öğrencilerin;

- RPL yönlendirme protokolünün temel prensiplerini (DODAG yapısı, rank kavramı, parent seçimi) kavramaları,
- Cooja simülatörü üzerinde bir IoT topolojisi kurabilmeleri, uygulamaları derleyip simülasyonları çalıştırabilmeleri,
- Flooding ve blackhole saldırılarının PDR (Packet Delivery Ratio), gecikme (latency) ve enerji tüketimi üzerindeki etkilerini gözlemleyip yorumlayabilmeleri,Saldırılara karşı savunma ve azaltma yöntemleri (ör. oran sınırlama, saldırı tespiti, güvenli RPL kullanımı) geliştirmeye yönelik fikirler üretebilmeleri,

beklenmektedir.

2.Genel Bilgiler

Bu bölümde laboratuvar deneyinde kullanılacak teknolojiler, protokoller ve saldırı türleri hakkında temel bilgiler verilmiştir.

Contiki / Contiki-NG

Contiki, kısıtlı bellek ve işlem gücüne sahip IoT cihazları için geliştirilmiş, açık kaynak kodlu, olay güdümlü bir işletim sistemidir. Contiki-NG (Next Generation) sürümü, IPv6, 6LoWPAN, RPL gibi IoT protokollerine destek sağlar. Cooja isimli simülatörü ile sanal düğümler oluşturulabilir, farklı ağ topolojileri denenebilir ve yönlendirme protokollerinin davranışları gözlemlenebilir.

6LoWPAN & RPL

- **6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks):** IEEE 802.15.4 standardı üzerine inşa edilen ve IPv6 paketlerini küçük MTU boyutuna sığdırabilmek için başlık sıkıştırma ve parçalama yapan uyarlama katmanıdır.
- **RPL (Routing Protocol for Low-Power and Lossy Networks):** IoT ağları için geliştirilmiş, yönlendirme kararlarını bir ağaç yapısı (DODAG

— Destination-Oriented Directed Acyclic Graph) üzerinden alan protokoldür.
    - Her düğümün bir rank değeri vardır; bu değer köke olan uzaklığı ve bağlantı kalitesini ifade eder.
    - Düğümler, kendilerinden daha düşük rank değerine sahip komşularını ebeveyn (parent) olarak seçerler.
    - Böylece tüm düğümler sink düğüme ulaşabilen çok atlamalı yollar kurar.

Flooding Saldırısı

Flooding saldırısında bir düğüm, protokolün öngördüğü hızın çok üzerinde paket üretir ve ağa gönderir. Bu saldırı sonucunda:

- Ağ bant genişliği gereksiz paketlerle dolar,
- Kuyruklar taşar ve normal paketler düşer,
- Paket kaybı artar, gecikmeler yükselir,
- Düğümlerin işlemci ve radyo bileşenleri sürekli çalıştığı için enerji tüketimi katlanır.

Blackhole / Sinkhole Saldırıları

- **Sinkhole:** Saldırgan düğüm, sahte düşük rank değeri yayınlayarak trafiği üzerine çeker.
- **Blackhole:** Trafiği üzerine çektikten sonra paketleri iletmez, tamamen yutar.
- Bu tip saldırılar özellikle çok atlamalı ağlarda uçtan uca teslim başarımını (PDR) büyük ölçüde düşürür.

Bu laboratuvarda uygulanacak blackhole saldırısı, kısa sürede gözlemlenebilir olması için basitleştirilmiş bir versiyondur: saldırgan düğüm belirli bir süre sonra radyo arayüzünü kapatarak üzerinden geçen trafiği “yutma” etkisi yaratır.

3.Ön Koşullar, Donanım/Yazılım

- Windows 10/11 yüklü PC, en az 8 GB RAM, 20 GB boş disk.
- VirtualBox (veya benzeri) + Ubuntu LTS (öneri: 22.04 veya 24.04).
- Ubuntu içinde: git, make, gcc, ant, OpenJDK (11 veya 17).
- Contiki-NG kaynak kodu ve Cooja.

Not: 60 dakikalık lab süresini verimli kullanmak için sanal makine ve paket kurulumlarının dersten önce yapılması şiddetle tavsiye edilir.

4.Zaman Planı (60 dk)

- 0–10 dk: VM'de Ubuntu oturumu, Contiki klasörüne geçiş, Cooja başlatma
- 10–20 dk: Temel topoloji (1 sunucu + 10 istemci) oluşturma, temel ölçümler
- 20–35 dk: Flooding saldırgan düğümü ekleme, gözlem/ölçüm
- 35–55 dk: Blackhole (basit) saldırganı ekleme, gözlem/ölçüm
- 55–60 dk: Hızlı değerlendirme ve rapor için veri notları

5.Deneyin Yapılışı (Adım Adım)

A) Windows → Ubuntu VM Kurulumu (özet)

- VirtualBox kurun, Ubuntu LTS ISO'su ile yeni VM oluşturun (4 GB RAM, 2 CPU yeterli).
- Ubuntu kurulumunu tamamlayın. "Guest Additions" opsiyonel.
- Ubuntu güncelleyin:
  sudo apt update && sudo apt -y upgrade

B) Gerekli Araçlar ve Contiki-NG kurulumu

 # Temel araçlar
sudo apt install -y git make gcc ant openjdk-11-jdk

 # Contiki-NG indirme
cd ~
git clone https://github.com/contiki-ng/contiki-ng.git
cd contiki-ng

C) Cooja'yı başlatma

cd tools/cooja
ant run

- İlk açılışta Cooja ana penceresi gelecektir.
- File → New simulation… ile yeni simülasyon başlatın (ör. isim: IoT-Sec-Lab).

D) Örnek Uygulamalar (RPL-UDP)

Contiki-NG *examples/ipv6/rpl-udp/* dizininde server ve client örnekleri bulunur. Bunları Cooja üzerinden derleyip mote olarak ekleyeceğiz.

Sunucu düğüm (sink) ekleme

- ***Motes*** → ***Add motes*** → ***Create new mote type*** → ***Cooja mote***
- ***Compile*** *penceresinde* ***Browse*** *ile:*
  `contiki-ng/examples/ipv6/rpl-udp/udp-server.c`
- ***Compile*** *(hedef:* `TARGET=cooja`*)* → ***Create*** → ***Add 1 mote****.*
- *Sunucuyu sahnenin merkezine yerleştirin.*

10 istemci düğüm ekleme

- *Aynı adımları izleyin, bu kez:*
  `contiki-ng/examples/ipv6/rpl-udp/udp-client.c`
- ***Create*** → ***Add 10 motes****.*
- *İstemcileri sunucunun etrafına yerleştirin.*

Radyo ortamı & bağlantı

- ***Simulation*** → ***Radio medium****: önce* ***UDGM Distance Loss*** *ile başlayın.*
- *Çok atlamalı rota görmek istiyorsanız* ***Directed Graph Radio Medium (DGRM)*** *seçip* ***Edit*** *ile el ile kenarları (bağlantıları) tanımlayabilirsiniz (Blackhole kısmında gerekli olacak).*

Log pencereleri ve ölçüm araçları

- ***Tools*** → ***Timeline****: Paket zaman çizelgesi*
- ***Tools*** → ***Radio messages****: Radyo iletileri*
- ***Tools*** → ***PowerTracker****: Enerji/Radyo çalışma yüzdeleri*
- ***View*** → ***Mote output****: Düğümlerin konsol çıktılarını görebilirsiniz.*
- *Sunucu çıktısında, gelen paket sayısı ve gönderen adresleri yazacaktır.*

Temel (saldırı yok) koşum

- ***Start*** *tuşu ile çalıştırın, 1–2 dakika gözlemleyin.*
- ***Not edin:*** *Toplam alınan paket, yaklaşık* ***PDR*** *(Packet Delivery Ratio), ortalama gecikme (Mote output zaman damgalarıyla kaba hesap),* ***PowerTracker*** *değerleri (ör. Radio ON yüzdesi).*

E) Flooding Saldırısı

Normal istemciler belirli aralıklarla veri yollar; saldırgan istemci bu aralığı çok küçülterek ağda tıkanma yaratır.

Kolay Uygulama (önerilen): İkinci bir istemci tipi oluşturun; saldırganın gönderim aralığını çok düşürün.

Saldırgan istemci kaynak dosyası hazırlığı
`udp-client.c` dosyasını kopyalayın (Ubuntu içinde):

```
cd ~/contiki-ng/examples/ipv6/rpl-udp
cp udp-client.c attacker_flood.c
```

`attacker_flood.c` içinde gönderim aralığını belirleyen makroyu küçültün. Örneğin (dosyada benzer bir tanım vardır):

```
#define SEND_INTERVAL (CLOCK_SECOND / 20) /* 50 ms: AŞIRI - flooding */
```

Not: Normal istemcilerde bu değer tipik olarak saniyeler mertebesindedir. 50–100 ms flooding etkisi için yeterlidir.

Cooja'da saldırganı ekleme

- **Motes → Add → Create new mote type → Cooja mote**
- **Kaynak: `attacker_flood.c`**
- **Create → Add 1 mote;** saldırganı sunucuya yakın konumlandırın.

Koşum ve gözlem

- Simülasyonu başlatın.
- **Sunucu Mote Output:** Gelen paketlerde sıçrama, kuyruk düşmeleri, kayıplar.
- **PowerTracker:** Radyo açık kalma yüzdelerinde artış.
- **Not edin:** Flooding varken/varken değil PDR ve gecikme kıyasları.

F) Blackhole Saldırısı (Basitleştirilmiş Yaklaşım)

Saldırgan düğüm kritik bir yerdeyken (bazı istemciler sunucuya yalnızca onun üzerinden ulaşabiliyorken) belirli bir andan sonra iletimi keser; o alt bölgedeki paketler "yutulur". Kısa süreli lab için, saldırganın radyo arayüzünü devre dışı bırakarak blackhole etkisi yakalanır.

1. **Topoloji hazırlığı (DGRM önerilir)**
    - **Radio medium → Directed Graph Radio Medium** seçin.
    - **Edit ile bazı istemciler → saldırgan → sunucu** olacak şekilde tek ara yol yaratın (yani o istemciler sunucuya yalnız saldırgan üzerinden ulaşsın).
    - Diğer istemciler doğrudan ya da başka yollarla bağlanabilir; böylece etkisi karşılaştırılabilir.

2. **Saldırgan blackhole uygulaması**
    - `examples/ipv6/rpl-udp` içinde yeni bir dosya oluşturun: `attacker_blackhole.c`
    - Aşağıdaki minimal süreç, başlangıçta normal düğüm gibi davranır; 60. saniyede radyo kapatır (paket iletimi biter → blackhole etkisi):

```
#include "contiki.h"
#include "net/ipv6/simple-udp.h"
#include "net/netstack.h"
#include "sys/log.h"

#define LOG_MODULE "BLACKHOLE"
#define LOG_LEVEL LOG_LEVEL_INFO

PROCESS(attacker_blackhole_process, "Attacker Blackhole (radio off after 60s)");
AUTOSTART_PROCESSES(&attacker_blackhole_process);

static struct etimer t;

PROCESS_THREAD(attacker_blackhole_process, ev, data)
{
  PROCESS_BEGIN();
  LOG_INFO("Blackhole attacker started; will turn radio OFF at t=60s\n");
  etimer_set(&t, 60 * CLOCK_SECOND);

  while(1) {
    PROCESS_WAIT_EVENT();
    if(etimer_expired(&t)) {
      LOG_INFO("Turning radio OFF now (blackhole active)\n");
      NETSTACK_RADIO.off();  /* iletim biter, gelen-giden paketler durur */
      /* Saldırgan kendisi de susar; RPL komşuları bir süre bu düğüme güvenmeye devam eder */
      PROCESS_EXIT(); /* süreç biter, düğüm radyo kapalı kalır */
    }
```

```
  }
  PROCESS_END();
}
```

- ***Cooja → Add mote*** *ile bu kaynaktan* ***1 saldırgan*** *oluşturun;* ***kritik bağlantı*** *konumuna yerleştirin.*

3. ***Koşum ve gözlem***

- ***Start*** *deyin;* ***t < 60 s*** *iken trafik normal akmalı (alt grup saldırgan üzerinden).*
- ***t ≈ 60 s*** *sonrası: o alt gruptan sunucuya paket akışı kesilecek; PDR düşecek, gecikmeler/timeoutlar görülecek.*
- ***Not edin:*** *Blackhole öncesi/sonrası PDR, gecikme, loglardaki zaman damgaları, PowerTracker değişimleri.*

Not: *Gerçek bir blackhole/sinkhole saldırganı, düşük* ***rank*** *yayınlayıp trafiği* ***bilerek üzerine çeker*** *ve sonra bırakır. Bu kısa labde, DGRM ile topolojiyi saldırganı "darboğaz" yapacak biçimde kurup, sonrasında* ***radyo kapatma*** *ile* ***benzer etkiyi*** *pratik ve hızlı biçimde gözlemliyoruz.*

6) Ölçüm, Metrikler ve Nasıl Not Alınır

PDR (Packet Delivery Ratio):

$$\text{PDR} = \frac{\text{Sunucuya ulaşan paket sayısı}}{\text{İstemcilerce gönderilen toplam paket}} \times 100$$

Gönderim periyotlarını biliyorsanız (ör. 10 istemci × 1 pkt/s) beklenen paketi tahmin edebilirsiniz.

Gecikme: *Sunucuda "paket zaman damgaları" ile kabaca ölçün (istemci gönderim logu varsa daha iyi).*
Enerji / Radyo: PowerTracker*'da saldırı ile* ***Radio ON*** *yüzdesindeki artışı gözleyin.*
RPL Davranışı: *Ebeveyn değişimleri, tekrar denemeler, loglarda uyarılar (varsa).*

7) Sık Karşılaşılan Sorunlar (Hızlı Çözüm)

Derleme hatası: openjdk-11-jdk ve ant kurulu mu? ant run komutunu tools/cooja klasöründe çalıştırın.
Hiç paket gelmiyor: Sunucu/istemci portları örnekte sabittir; doğru örnek dosyaları seçtiğinizden emin olun (udp-server.c / udp-client.c).
Çok atlama yok: Düğümler çok yakınsa tek atlama olur; DGRM açıp Edit ile elle kenarları ayarlayın.
Blackhole etkisi zayıf: Saldırgan gerçekten darboğaz mı? DGRM ile o alt grubun tek çıkışını saldırgan yapın.
Flooding etkisi yok: SEND_INTERVAL yeterince küçük mü (ör. CLOCK_SECOND/20)? Saldırganı sunucuya yakın yerleştirin.

8) Hazırlık Soruları

1. 6LoWPAN ve RPL nedir? RPL'de rank neyi ifade eder?
2. RPL'de ebeveyn (parent) seçimi hangi metriklerle yapılabilir? (ör. ETX)
3. Flooding saldırısı ağda hangi kaynakları tüketir ve sonuçları nelerdir?
4. Blackhole/Sinkhole saldırıları arasındaki fark nedir?
5. PDR nasıl hesaplanır, hangi durumlarda hızla düşer?
6. Enerji tüketimi neden güvenlik açısından da kritiktir?
7. RPL'de Trickle mekanizmasının amacı nedir? Saldırılar bunu nasıl etkileyebilir?
8. Flooding'e karşı oran sınırlama (rate limiting) nasıl yardımcı olur?
9. RPL'in güvenli modları veya katmanda 802.15.4 AES-CCM kullanımı ne sağlar?
10. Cooja'da DGRM kullanmanın ölçüm açısından avantajları nelerdir?

9) Raporlama Yönergesi

Teslim: Deneyden sonra tek PDF (en fazla ~6–8 sayfa). Aşağıdaki başlıkları içermelidir.

9.1. Deney Tasarlama

- Kurulan topoloji (şema veya ekran görüntüsü), düğüm sayıları/rolleri.
- Radio medium seçimi (UDGM/DGRM) ve gerekçesi.

- Flooding ve Blackhole senaryolarının parametreleri (ör. **SEND_INTERVAL**, blackhole tetik zamanı).

9.2. Deney Yapma

- Adımların özeti (Cooja kurulumu, derlemeler, simülasyon başlatma).
- Normal çalışma, flooding ve blackhole için ayrı koşumlar (en az 1–2 dakika).
- Kullanılan ölçüm araçları (Timeline, Radio messages, PowerTracker, Mote output).

9.3. Veri Toplama

- Her senaryo için **PDR**, ortalama/tepe **gecikme** (kaba hesap kabul), **PowerTracker**'dan radyo çalışma yüzdesi.
- Kritik gözlemler (ör. sunucu logunda kuyruk hataları, yönlendirme değişimleri).

9.4. Sonuçları Analiz Etme ve Yorumlama

- Normal ↔ Flooding ↔ Blackhole sonuçlarının **tablo**/grafiklerle karşılaştırması.
- Flooding'de ağ doygunluğu ve bunun PDR/enerjiye etkisi.
- Blackhole'de alt ağın etkilenme biçimi, yeniden-yönlendirme gecikmeleri.
- **Sınırlamalar:** Basitleştirilmiş blackhole yaklaşımı ile gerçek sinkhole/blackhole farkları.
- **Öneriler:** Oran sınırlama, saldırı tespiti (anormallik tabanlı), güvenli RPL/anahtar yönetimi, çoklu kök veya yedek yol fikirleri.

Ekler (Opsiyonel):

- Kullandığınız `attacker_flood.c`, `attacker_blackhole.c` ve/veya yaptığınız küçük kod değişiklikleri.
  DGRM bağlantı ekran görüntüsü.
- Kısa log parçaları.

10) Ek: Örnek Kod Parçaları

10.1. Flooding Saldırgan (istemci varyantı)

examples/ipv6/rpl-udp/attacker_flood.c (temel fark **SEND_INTERVAL**):

```
#include "contiki.h"
#include "net/ipv6/simple-udp.h"
#include "sys/log.h"
#define LOG_MODULE "FLOOD"
#define LOG_LEVEL LOG_LEVEL_INFO

#define UDP_PORT 8765
#define SEND_INTERVAL (CLOCK_SECOND / 20) /* 50 ms */

static struct simple_udp_connection udp_conn;
PROCESS(attacker_flood_process, "Flooding attacker");
AUTOSTART_PROCESSES(&attacker_flood_process);

PROCESS_THREAD(attacker_flood_process, ev, data)
{
  static struct etimer periodic_timer;
  static uip_ipaddr_t dest_addr;
  static uint32_t seq;

  PROCESS_BEGIN();

  /* Sunucu portu ve adresi: RPL ile köke gidecek; hedefi
::1 benzeri kök adresi yerine,
     simple_udp_sendto'da NULL hedefle de RPL yönlendirme
uygulanır. */
       simple_udp_register(&udp_conn,   UDP_PORT,   NULL,
UDP_PORT, NULL);

  etimer_set(&periodic_timer, SEND_INTERVAL);
```

```
  while(1) {

PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer))
;
    etimer_reset(&periodic_timer);
    char buf[32];
      int len = snprintf(buf, sizeof(buf), "FLOOD %lu",
(unsigned long)seq++);
    simple_udp_sendto(&udp_conn, buf, len, NULL);
      LOG_INFO("Flood packet sent, seq=%lu\n", (unsigned
long)seq);
  }

  PROCESS_END();
}
```

10.2. Blackhole (Basitleştirilmiş – 60. sn'de radyo kapat)

examples/ipv6/rpl-udp/attacker_blackhole.c

```
#include "contiki.h"
#include "net/netstack.h"
#include "sys/log.h"

#define LOG_MODULE "BLACKHOLE"
#define LOG_LEVEL LOG_LEVEL_INFO

PROCESS(attacker_blackhole_process, "Blackhole attacker
(radio off at 60s)");
AUTOSTART_PROCESSES(&attacker_blackhole_process);

PROCESS_THREAD(attacker_blackhole_process, ev, data)
```

```
{
  static struct etimer t;
  PROCESS_BEGIN();
    LOG_INFO("Running  normal  until  t=60s,  then  radio
off\n");
  etimer_set(&t, 60 * CLOCK_SECOND);

  while(1) {
    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&t));
       LOG_INFO("Radio  OFF  ->  packets  will  be  dropped
(blackhole effect)\n");
    NETSTACK_RADIO.off();
    PROCESS_EXIT();
  }
  PROCESS_END();
}
```