Fakultet tehničkih nauka, Univerzitet u Novom Sadu

Predmet: Informaciona bezbednost u infrastrukturnim sistemima (IBIS)

Tema projekta: Bezbednost računarskih mreža u KI

Student: Nikola Miljković

Datum: Oktobar 2025

## 1. Uvod

Ovaj projekat je edukativni lab simulator za istraživanje sigurnosti industrijskih kontrolnih sistema (ICS/OT), posebno:

* demonstrira kako Modbus TCP komunikacija između HMI/klijenta i PLC/server izgleda,
* pokazuje kako man-in-the-middle napad može izmeniti očitane vrednosti (bez stvarnog ARP spoofinga — koristi se eksplicitni proxy),
* daje interaktivni UI (Flask) koji prikazuje očitanu vrednost i omogućava uključivanje/isključivanje manipulacije u vreme rada (control.json),
* sadrži opcionalni defense modul (ARP detektor) i pomocni alat za autorizovano pisanje u registar (manipulator).

Cilj: studentima omogućiti razumevanje protokola, napada (MITM), učinke promena u fizičkom procesu (simuliranom temperaturom), praćenje logova i osnovnu obradu incidenta u kontrolisanom okruženju

# 2. Komponente

1. modbus\_server.py — simulirani PLC / Modbus TCP server. Svakih N sekundi generiše novu vrednost temperature i zapisuje je u holding register 0.
2. mitm\_proxy.py — eksplicitni Modbus proxy: čita registre sa stvarnog servera, po potrebi manipuliše (poveća reg0 za +10), zatim postavlja vrednosti u DataBank (proxy server predstavlja HMI klijentu).
3. modbus\_client.py — HMI / operator client koji periodično čita register 0 s proxyja i loguje prikazanu temperaturu.
4. mitm\_modbus\_manipulator.py — pomoćni alat da autorizovano upiše vrednost u registar (test-only).
5. ui\_server.py — Flask UI koji prikazuje trenutnu vrednost (čita preko proxyja), prikazuje spojene logove, služi kao kontrolni panel: ON/OFF manipulacije (piše u control.json) i omogućava autorizovan upis u registar.
6. defense\_module.py — ARP anomaly detector (scapy) — detektuje promene MAC <-> IP mappinga (gratis ARP / promjena MAC) — koristi se kao primer mrežnog odbrambenog alata.
7. run\_with\_ui.sh / run\_local.sh / stop scripts / docker-compose — skripte za pokretanje okruženja lokalno ili u Dockeru.
8. control.json — jednostavan JSON fajl kojim UI i proxy koordiniraju: {"manipulate": true/false, "ts": ...}.

# 3. Koji protokoli se koriste i zašto

* TCP (Transport Layer) — Modbus TCP radi preko TCP konekcija.
* Modbus TCP (aplikacioni protokol) — standardni industrijski protokol koji koristi registarske adrese (holding registers, input registers, coils). Ovde koristimo read\_holding\_registers i write\_single\_register.
* HTTP (Flask UI) — UI server služi HTML stranice i API endpoint /api/value.
* (Opcionalno) ARP - scapy — defense\_module sniffa layer2 ARP poruke (gratuitous ARP i promene MAC adresa).

Napomena: Ne vršimo ARP spoofing u projektu (mitm\_attack.py je stub iz sigurnosnih razloga). MITM se simulira proxyjem — eksplicitna aplikacija koja zastupa server prema klijentu.

# 4. Kako radi — detaljan tok podataka (korak po korak)

1. Server (PLC):
   1. Pokreće Modbus TCP server (pyModbusTCP.ModbusServer).
   2. Svakih interval sekundi generiše nasumičnu vrednost temperature (npr. 20–100 °C).
   3. Tu vrednost upisuje u holding register 0 (DataBank.set\_words(0, [temperature])).
   4. Loguje: [SERVER] Temperature set to X °C (holding register 0)
2. Proxy (MITM simulator):
   1. Pokreće Modbus server endpoint na npr. 0.0.0.0:1502 (to je endpoint za HMI/klijenta).
   2. Paralelno održava Modbus klijenta prema stvarnom serveru (server\_host:server\_port).
   3. U petlji (sync loop) povlači "read\_holding\_registers(0, 10)" s pravog servera.
   4. Ako control.json (ili argumenti) kažu manipulate: true, proxy mijenja regs[0] = regs[0] + 10 (ili neku drugu logiku). To ilustruje lažno očitanje temperature.
   5. Nove (kopirane/manipulisane) vrednosti upisuje u svoje DataBank tako da HMI klijent vidi izmenjene vrednosti.
   6. Log: [PROXY] Manipulated reg0: <orig> -> <new> (ako manipulacija uključena).
3. Client (HMI):
   1. Pokušava se povezati na proxy (npr. 127.0.0.1:1502).
   2. Svakih poll\_interval sekundi čita read\_holding\_registers(0,1).
   3. Ispisuje log: [CLIENT] Current temperature read from register 0: X °C — ovo je vrednost koju HMI vidi (može biti man-in-the-middle izmenjena).
4. UI (Flask):
   1. UI pruža endpoint / i /api/value.
   2. /api/value čita trenutnu vrednost preko Modbus klijenta prema proxyju (isto kao HMI).
   3. UI prikazuje:
      1. trenutnu vrednost (register 0),
      2. logs aggregated iz logs/ foldera (modbus\_server.log, modbus\_proxy.log, modbus\_client.log),
      3. dugme za toggle koje menja control.json ( uključuje/isključuje manipulaciju),
      4. formu za autorizovano upisivanje vrijednosti putem mitm\_modbus\_manipulator.py.
   4. UI ne izvodi manipulaciju — samo menja control.json i poziva manipulator ako treba.
5. control.json:
   1. Proxy periodično (u svakom sync ciklusu) proverava ovaj fajl i ažurira self.manipulate prema {"manipulate": true/false}.
   2. UI piše u isti fajl kada korisnik pritisne toggle.
6. Defense module (opc.):
   1. Pokreće sniffing na mrežnom interfejsu i izbacuje upozorenje ako IP mapping promeni MAC (mogući ARP spoofing). Ovo je odvojeno od proxy mehanike i služi kao demonstracija detekcije mrežnog napada.

# 5. Šta projekt meri i pokazuje — cilj merenja

* Merenje: simulira i prikazuje vrednost procesa (temperatura) koju PLC generiše i HMI očitava.
* Pokaz: učinak MITM (manipulacija podacima) — kako napadač može izmeniti očitanu vrednost bez da PLC zna.
* Razlika stvarno vs. prikazano: server (stvarna vrednost) loguje pravu temperaturu; client (HMI) i UI prikazuju vrednost koju proxy posreduje (može biti lažna).
* Audit/logging: logovi (server/proxy/client/ui) služe za forenzičku analizu; proxy log pokazuje koji su zapisi manipulirani.
* Kontrola & autorizacija: mitm\_modbus\_manipulator služi kao "autorizovan" unos (npr. održavanje) — pokazuje razliku između legitimnog pisanja i neautorizovane manipulacije.

# 6. Kako pokrenuti lokalno (korak-po-korak)

## A) Bez Docker-a — skripte (na tvojoj mašini)

1. Osiguraj Python 3.8+ i python3 -m venv .venv
2. Aktiviraj venv: source .venv/bin/activate (mac/linux) ili .\.venv\Scripts\activate (Windows PowerShell)
3. Instaliraj: pip install -r requirements.txt
4. Pobrinite se da imate control.json u root projektu (ako nema, napravi):

{"manipulate": true, "ts": 0}

1. Pokreni skriptu koja starta sve sa UI:

chmod +x run\_with\_ui.sh ./run\_with\_ui.sh 15002 1502 8080

Ovo će startati server na localhost:15002 (ili server host/port koji daš), proxy na :1502 i UI na :8080 (ako koristiš iste argumente).

1. Otvori UI: <http://127.0.0.1:8080>
2. Logovi: tail -f logs/modbus\_server.log logs/modbus\_proxy.log logs/modbus\_client.log logs/ui\_server.log
3. Zaustavi sve: ./stop\_local\_ui.sh ili ./stop\_local.sh (ovisno koju skriptu koristiš).

B) Preko Docker Compose

Napravi control.json u root (kao gore).

Build + up:

UI\_PORT=8080 docker-compose up --build -d

Ako host već koristi 8080 (npr. Jenkins), pokreni na drugom:

UI\_PORT=8082 docker-compose up --build -d

Pogledaj log:

docker-compose logs -f ui\_server modbus\_proxy modbus\_server modbus\_client

# Kako koristiti UI i control.json (praktčno)

* Otvori UI u browseru.
* Gornji panel prikazuje trenutnu očitanu vrednost (preko proxyja).
* Manipulacija: pritisni Disable/Enable da promeniš control.json. UI će upisati manipulate: false ili true. Proxy će na idućem ciklusu pročitati control.json i prema tome uključiti/isključiti manipulaciju.
* Direktno pisanje: polje Direktno upisivanje koristi mitm\_modbus\_manipulator.py na serveru — to je legitiman način da operator upiše vrednost u registar (može poslužiti za test).
* Logs: UI agregira i prikazuje poslednje linije iz server/proxy/client logova.

# 8. Kako se napad "manifestuje" u ovom labu

* Napad se simulira tako što proxy, koji stoji između HMI i PLC, izmijeni očitanu vrijednost.
* U stvarnom svijetu MITM napad tipično uključuje ARP spoofing, BGP hijack ili kompromitovan ruter. Ovdje, u složenijoj varijanti, proxy ne radi ARP spoofing — to je sigurnosna odluka. Umjesto toga:
* vi namjerno konfigurirate HMI da se poveže na proxy (ili u testbed-u ARP spoofing može biti izveden u kontrolisanom mrežnom labu).
* proxy zatim čita pravu vrijednost, mijenja je i servira lažni rezultat HMI-ju.
* Posljedice: operator vidi pogrešnu temperaturu i može donijeti pogrešne odluke (npr. ne pokrene alarm, dozvoli opasno stanje).

# Šta proxy radi (tehnički)

* Dva modula u proxy-u:
  + ModbusClient — povezuje se na stvarni server i čita registere.
  + ModbusServer — izlaže endpoint za HMI/klijenta i drži DataBank vrednosti koje HMI čita.
* Synchronization loop:
  + client.read\_holding\_registers(0,10)
  + kopira listu registara
  + pročita control.json (ako postoji) i postavi self.manipulate na boolean iz njega
  + ako manipulate True → izmeni kopirani registar 0 (npr. +10)
  + DataBank.set\_words(0, copied)
* Na taj način HMI-ju izgleda kao da su vrednosti od servera (ali su zapravo proxy-jeve).

# Šta client radi

* Jednostavan Modbus TCP klijent.
* Pokušava se periodično (svakih N sekundi) povezati i čitati holding register 0.
* Loguje rezultat (što je ono što operator vidi).
* U produkciji HMI bi imalo GUI, alarmne pragove i istoriju— ovde je pojednostavljeno.

# 10. Šta server radi

* Simulira fizički senzor (temperaturu).
* Generiše slučajne vrednosti unutar raspona.
* Ako bi ovo povezali sa stvarnim PLC-em — PLC bi umesto generatora čitao fizički senzor.

# 11. Sigurnosne napomene (bitno za lab i ocjenu)

* Ne izvoditi ARP spoofing na produkcijskoj mreži ili bilo kojem mrežnom segmentu gde nemate dozvolu.
* Ovaj lab sadrži alate koji pokazuju efekt MITM, ali ne sadrži automatski štetan kod.
* Ako želite vežbati ARP spoofing, koristite izolovan lab (VM, Mininet, GNS3) i obavezno upotrebite defense\_module.py za detekciju i nadzor.

# 12. Kako ovo pokriva teme iz kursa IBIS

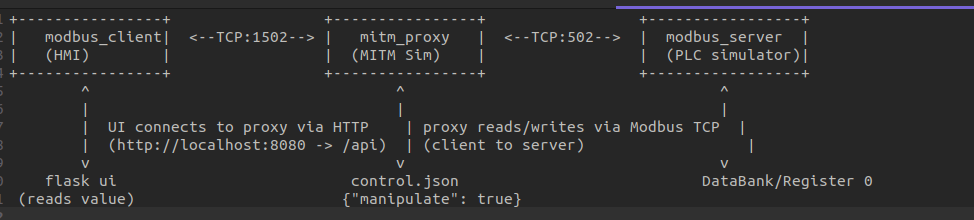
* Analiza rizika i napada (predavanja i vežbe) — demonstrira realan napad na OT protokol (Modbus TCP) i kako to utiče na odluke u HMI.
* Mrežna analiza (wireshark/tcpdump) — omogućava hvatanje i analizu paketa između klijenta, proxyja i servera (vidljivi su Modbus TCP request/response).
* Sistemi sistema / SCADA — simulira delove topologije: PLC, proxy (mrežni element), HMI i monitoring.
* Reakcija na incident — logovi i opcionalni defense modul podržavaju otkrivanje i forenziku.
* Standardi i protokoli — praktično razumevanje Modbus protokola i njegovih slabosti (lack of authentication).

# 14. Predloženi eksperimenti (za lab zadatke)

1. On/Off manipulacija:
   1. Uključi manipulate: true i uporedi stvarnu temperaturu (server log) i prikazanu tempaturu (client/UI).
   2. Izračunaj bias i uticaj (npr. kada vrednost prelazi kritičnu granicu?).
2. Vremenski napad:
   1. Neka proxy manipuliše samo u određenim intervalima ili prilikom određenih vrednosti (npr. manipuliši samo ako temperatura ∈ [30,40]).
   2. Istraži kako bi alarmi reagovali.
3. Detekcija:
   1. Pokreni defense\_module.py na istom segmentu i simuliraj ARP spoofing u izolovanom labu da vidiš alert.
   2. Snimi pcap i uporedi stvarne vs lažne pakete.
4. Forenzika:
   1. Analiziraj logs i pcap da odrediš da li je izmena bila automatska (proxy) ili je došla iz lošeg PLC-a.
5. Autentifikacija:
   1. Razvij unapređenje: dodaj HMAC/bitove integriteta u komunikaciju između HMI i PLC i pokaži kako to sprečava proxy izmene (u testnom protokolu).

# 15. Dijagrami

### 1) ASCII network overview



### 2) Mermaid sequence diagram

sequenceDiagram

participant UI as Flask UI

participant HMI as Modbus Client (HMI)

participant Proxy as Modbus Proxy

participant PLC as Modbus Server (PLC)

UI->>Proxy: GET /api/value (reads reg0)

Proxy->>PLC: Modbus Read(reg0..)

PLC-->>Proxy: reg0 = 34

alt manipulate true

Proxy-->>Proxy: reg0 <- reg0 + 10 (manipulated -> 44)

end

Proxy-->>UI: JSON {value: 44}

HMI->>Proxy: Modbus Read(reg0)

Proxy-->>HMI: reg0 = 44

Primer control.json

{

"manipulate": true,

"ts": 1696560000

}

* manipulate: true → proxy manipuliše reg0 (+10).
* UI će menjati manipulate klikom na dugme.

# 16. Logovi i šta tražiti u njima (forenzika)

* modbus\_server.log — stvarne temperature servera.
* modbus\_proxy.log — koji su reg0 orig -> new (pokazuje manipulacije).
* modbus\_client.log — šta HMI vidi.
* ui\_server.log — requesti i errori UI; može otkriti probleme (npr. TemplateNotFound ako templates nisu montirani).

Ako vidiš:

* Server=33, Proxy manipulated reg0: 33 -> 43, Client reads 43 → jasno MITM.

# 17. Zaključak — glavna poenta projekta

* Edukativna demonstracija osnovnih slabosti industrijskih protokola (Modbus) i kako man-in-the-middle može menjati vrednosti senzora bez znanja PLC-a.
* Omogućava studentima da vežbaju otkrivanje, praćenje i forenziku napada (logovi + defense module).
* Pruža UI za kontrolu i brz vizualni pregled, što olakšava labske zadatke i ocenu.
* Projekt je praktičan primer kako koncepti iz kursa (tipovi napada, analiza mrežnog saobraćaja, incident response, ICS protokoli) funkcionišu u stvarnom (simuliranom) okruženju.