
UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Laurea Triennale in Informatica

# Honeypot per Sistemi di Controllo Industriale (ICS)

Relatori:

**Prof. Fabrizio Baiardi**

**Prof. Luca Deri**

Candidata:

**Lorena Modica**

ANNO ACCADEMICO 2023/2024

**Abstract**

Negli ultimi anni anche i sistemi industriali sono stati connessi alle reti aziendali e Internet. Questo se da una parte ha apportato enormi benefici, dall'altra ha esposto questi sistemi a dei seri rischi.
Uno dei protocolli più longevi è Modbus. Essendo nato per ambienti isolati non sono stati implementati controlli per la sicurezza, quindi risulta vulnerabile sotto diversi punti di vista.
Un metodo che può essere sfruttato per ovviare a questo problema potrebbe essere l'uso di honeypot: sistemi che hanno come obiettivo quello di ingannare un utente malevolo.
In questo studio viene preso in esame Conpot: un honeypot a bassa interazione che raccoglie in un file di log le informazioni riguardo le interazioni. Ci si pone come obiettivo quello di cercare di profilare dei possibili attacchi nei confronti di dispositivi Modbus, facendo delle statistiche e studiando eventuali pattern che vengono reiterati nel corso delle interazioni.

## Risultati raggiunti

Durante questo lavoro di tesi sono stati raggiunti alcuni obiettivi significativi. In primo luogo è stata fatta una mappatura sulla distribuzione degli attacchi per aree geografiche, per giorni della settimana e per fasce orarie. Sono poi state raccolte informazioni sugli attaccanti, sul loro modo di interagire con l'honeypot e sulla natura degli IP collezionati. Dopodiché è stato possibile fare delle assunzioni sugli attacchi ponendo l'attenzione sulla durata delle sessioni, ricercando comportamenti simili e valutando se fossero stati fatti manualmente o in maniera automatica.

# Indice


**1 Introduzione** **9**

**2 Sistemi per il controllo industriale: ICS, OT e SCADA** **11**
2.1 Operational Technology - OT . . . . . 12
2.1.1 RTU e PLC . . . . . 13
2.2 Industrial Control Systems - ICS . . . . . 13
2.3 Supervisory Control and Data Acquisition - SCADA . . . . . 15

**3 Sicurezza nei sistemi ICS e protocolli industriali** **17**
3.0.1 STUXNET . . . . . 18
3.0.2 Statistiche recenti sugli attacchi ai sistemi industriali . . . . . 18
3.1 Protocolli industriali . . . . . 19
3.1.1 Ethernet/IP . . . . . 20
3.1.2 Profinet IO . . . . . 20
3.1.3 MTCONNECT . . . . . 20
3.1.4 OPC-UA . . . . . 21
3.1.5 Modbus . . . . . 21
3.2 Modbus TCP . . . . . 25
3.2.1 ADU Modbus su TCP/IP . . . . . 26
3.2.2 Vulnerabilità del protocollo Modbus TCP . . . . . 27

**4 Generalità degli Honeypot** **31**
4.1 Honeypot . . . . . 32
4.1.1 HoneyD . . . . . 32
4.1.2 Capture-HPC . . . . . 33
4.1.3 HoneyPLC . . . . . 33
4.1.4 LOGistICS . . . . . 33
4.1.5 GasPot . . . . . 34
4.1.6 Gridpot . . . . . 34
4.1.7 Conpot . . . . . 34
4.2 Architettura di riferimento di un Honeypot Modbus in ambito SCADA 35
4.3 Honeypot a confronto . . . . . 36
4.3.1 Considerazioni sulla scelta finale dell'honeypot Conpot . . . . 36

**5 Testing, raccolta dati e analisi utilizzando Conpot 39**
5.1 Prerequisiti . . . 39
5.1.1 Docker . . . 39
5.2 Test preliminari . . . 39
5.2.1 Installazione . . . 39
5.2.2 Personalizzazione . . . 40
5.2.3 Fase di testing . . . 41
5.3 Analisi dei dati . . . 42
5.3.1 Obiettivo . . . 42
5.3.2 Deployment . . . 43

**6 Considerazioni finali 59**
6.0.1 Conclusioni . . . 62
6.0.2 Sviluppi Futuri . . . 63

# Capitolo 1

# Introduzione

Negli ultimi anni ragioni di efficienza e di economicità hanno reso sempre più popolari sistemi di controllo industriali connessi sia alle reti aziendali che ad Internet. Da un lato questo ha prodotto benefici economici non trascurabili ma dall'altro ha anche generato rischi significativi, poiché i sistemi di controllo industriale possono essere attaccati da remoto via Internet.
Uno degli esempi più rilevanti di questo problema è il protocollo Modbus. Questo protocollo in origine è stato progettato per ambienti isolati e quindi non adotta meccanismi classici per la sicurezza, quali autenticazione o protezione dei messaggi scambiati. Di conseguenza Modbus presenta numerose vulnerabilità, questo spiega perché i sistemi che lo utilizzano siano spesso bersagli di intrusioni informatiche.

Questa tesi vuole raccogliere ed esaminare dati su queste intrusioni. La soluzione adottata è basata su honeypot, sistemi che hanno come obiettivo quello di ingannare un utente malevolo.

In particolare, il sistema per la raccolta dei dati in seguito presentati è basato su Conpot: un honeypot a bassa interazione che raccoglie e memorizza in un file di log le informazioni sugli attacchi subiti.

Obiettivo della tesi è di profilare i possibili attacchi nei confronti di dispositivi di controllo industriale che utilizzano Modbus, producendo delle statistiche e ricercando pattern che vengono reiterati nel corso delle interazioni tra l'honeypot e gli attaccanti.

Questo lavoro è cosi organizzato:

- Nel capitolo 2 verrà introdotto il dominio del discorso.

- Nel capitolo 3 sarà brevemente descritto il caso reale STUXNET. In seguito verrà fatta una breve presentazione dei protocolli industriali con un approfondimento del protocollo Modbus TCP. Verrà spiegata la differenza tra il protocollo Modbus seriale e TCP. Infine, verranno descritti i pacchetti Modbus TCP e le vulnerabilità tipiche di questo protocollo.

- Nel capitolo 4 verrà esaminato il concetto di honeypot in generale ed analizzato, in particolare, il funzionamento di Conpot.
- Nel capitolo 5 verranno presentati i dati raccolti con i conseguenti attacchi all'honeypot registrati.
- Nel capitolo 6 verranno analizzati i dati raccolti con l'intento di profilare alcuni comportamenti tipici degli attaccanti in ambito ICS.

# Capitolo 2

# Sistemi per il controllo industriale: ICS, OT e SCADA

Per parlare dell'uso degli honeypot nel mondo **ICS** *(Industrial Control Systems)* è necessario introdurre e differenziare alcuni termini. Gli acronimi **OT** *(Operational Technology)*, ICS e **SCADA** *(Supervisory Control and Data Acquisition)* sono correlati ma non sono la stessa cosa [1]-[2]-[3].
La figura 2.1 descrive come sono relazionati questi componenti tra loro.

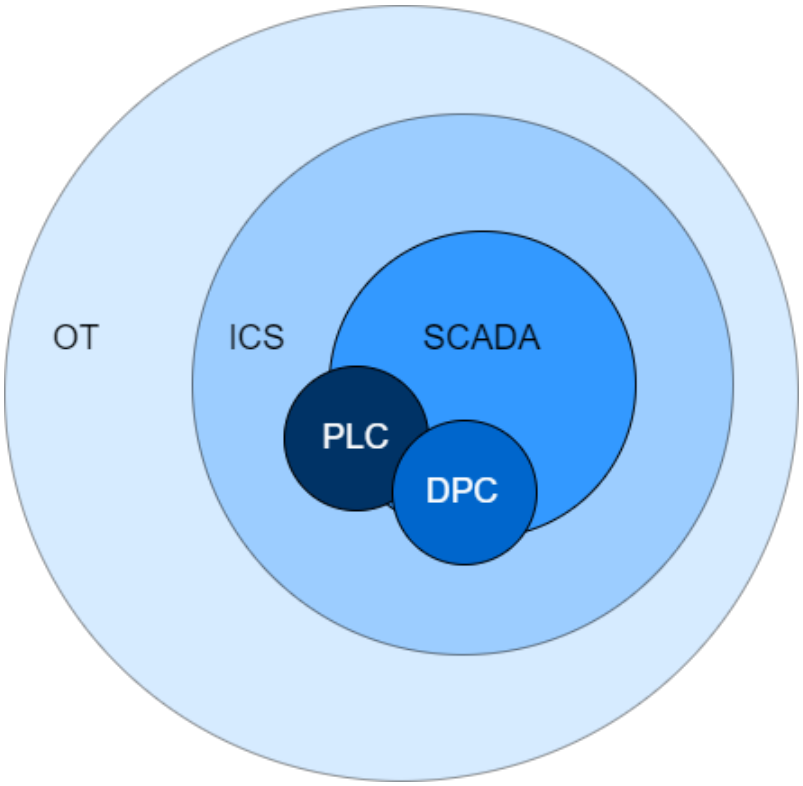


Figura 2.1: Relazione tra OT, ICS, SCADA [1]

## 2.1 Operational Technology - OT

Il termine OT descrive sistemi informatici utilizzati per gestire le operazioni industriali (gestione delle linee di produzione, operazioni di monitoraggio, etc.) tramite processi e dispositivi fisici [3]. Questi ambienti solitamente contengono:

- Sistemi ICS (e.g., **SCADA**)
- Sistemi di controllo di processo discreti (DPC [1])
- Controllori logici programmabili (**PLC**) e/o unità terminali remote (**RTU**) che verranno approfonditi nel prossimo paragrafo
- Reti e unità organizzative dedicate

[1] *Discrete Process Control Systems*: questi sistemi utilizzano controllori logici programmabili o altri dispositivi di controllo.

### 2.1.1 RTU e PLC

I PLC sono dispositivi industriali che ricevono informazioni sulle condizioni di un processo e le inviano ai dispositivi di un sito di produzione per controllare gli impianti. Interpretano segnali e generano variabili da inviare agli attuatori [2][4]. Sono dotati di una CPU in grado di elaborare i dati e vengono utilizzati per il controllo in aree locali.
Le RTU sono *field devices* che vengono posizionate vicino al processo da monitorare o gestire e in seguito inviano le informazioni a un sistema SCADA. Sono adatti per una distanza geografica ampia poiché utilizzano la comunicazione wireless. La loro capacità di elaborazione dati può essere nulla o minima [5]-[6]-[1]-[7].
La tabella nella figura 2.2 mostra le differenze sostanziali tra questi due dispositivi.

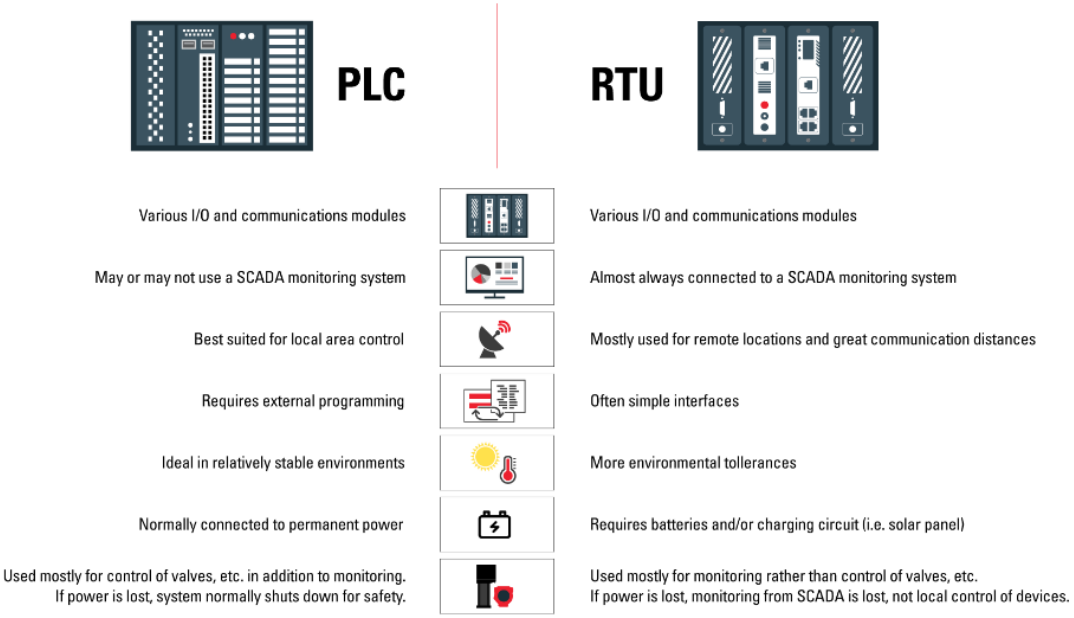


Figura 2.2: PLC e RTU a confronto [6]

## 2.2 Industrial Control Systems - ICS

L'ICS (figura 2.3) è un sottinsieme del settore dell' *Operational Technology* che comprende i sistemi utilizzati per monitorare e controllare i processi industriali. Esempi sono gli impianti di produzione di energia elettrica, sistemi di trasporto, raffinerie di petrolio, fabbriche chimiche e, in generale, impianti di produzione di vario tipo. L'*Industrial Control Systems* è in genere un sistema essenziale per la sopravvivenza

[2]Ad esempio valvole di controllo, interruttori, commutatori e motori. Sono utilizzati per manipolare direttamente il processo in base ai comandi del controllore.

di un'azienda o di un organizzazione, motivo per cui uno dei requisiti che lo caratterizza è quello dell' *high-availability*, ovvero la capacità di non avere dei singoli punti di guasto affinché in casi critici non venga compromesso il funzionamento dell'intero sistema.
La maggior parte degli ICS rientra nella categoria dei sistemi cui le operazioni svolte vengono monitorate in maniera continuativa *(continuous process control system)*, questo aspetto è tipicamente gestito tramite PLC [4]-[8].

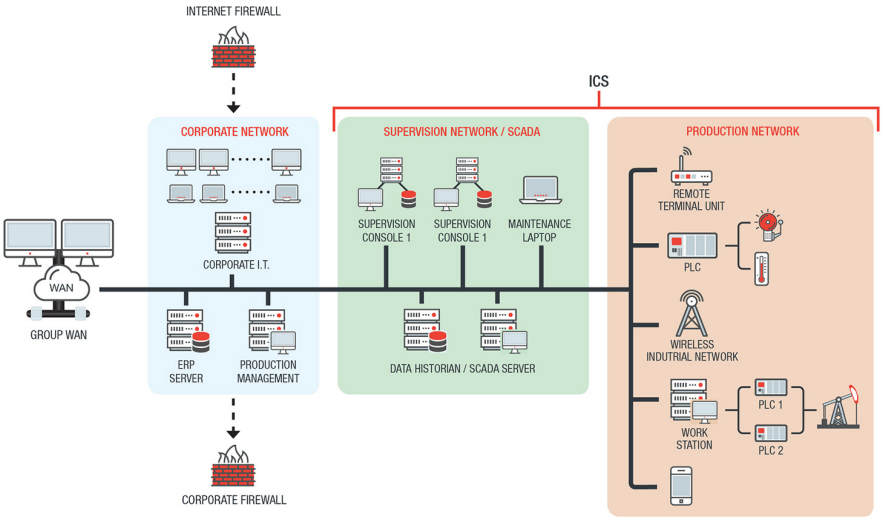


Figura 2.3: Sistema ICS, Trend [9]

## 2.3 Supervisory Control and Data Acquisition - SCADA

L' obiettivo principale del *Supervisory Control and Data Acquisition* è il monitoraggio e il controllo dello stato di un processo attraverso un sistema centralizzato che permette l' automazione di compiti, come la raccolta dati dapprima gestiti dagli operatori (figura 2.4).

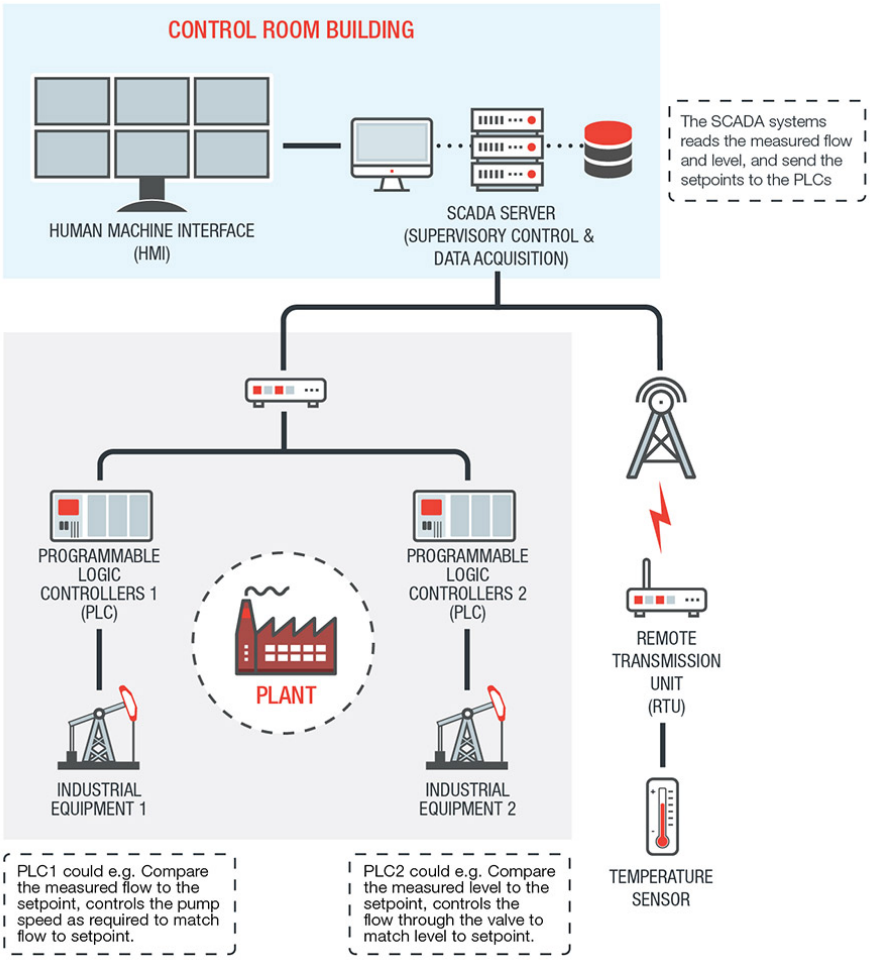


Figura 2.4: Funzione di un sistema SCADA, Trend [9]

I *field devices* (e.g., RTU) controllano le operazioni locali, come l'apertura o la chiusura di valvole e interruttori, la raccolta di dati dai sensori e il monitoraggio dell'ambiente locale per rilevare eventuali condizioni anomale.
I sistemi SCADA visualizzano messaggi di stato e allarmi tramite un display ed inoltre mettono a disposizione degli operatori funzionalità che consentono di modificarne il funzionamento senza percorrere lunghe distanze [4].
Come raffigurato nella figura 2.5 i componenti principali di un sistema SCADA, oltre ai già citati display, PLC e RTU sono:

- ***Control unit***: unità che si occupa di collegare le *remote terminal unit* al sistema SCADA e che ne permette lo scambio di dati da e verso le unità terminali, in tempo reale e quindi con bassa latenza.

- ***Communication link***: I collegamenti di comunicazione possono essere di tipo Ethernet per un sistema di produzione, collegamenti WAN su Internet o radio per operazioni distribuite oppure collegamenti telemetrici per le apparecchiature che si trovano in un'area remota senza strutture di comunicazione.

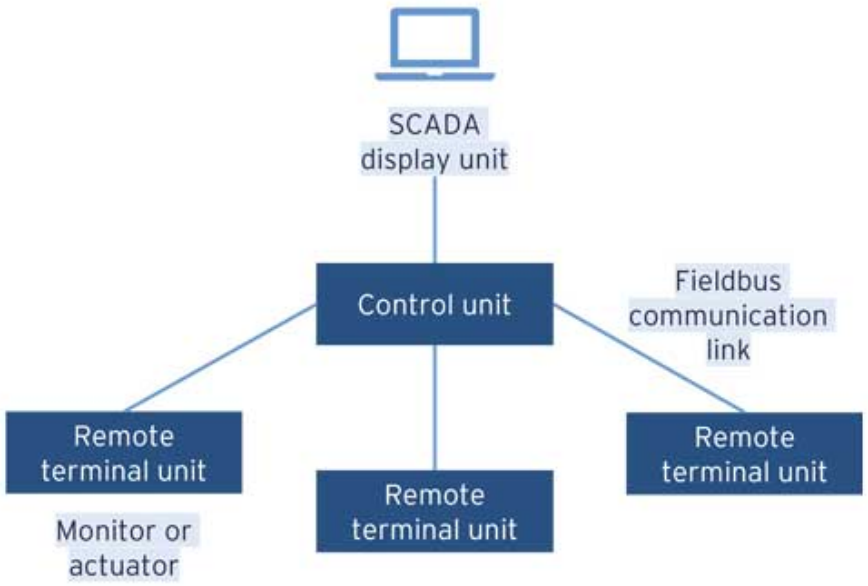


Figura 2.5: Tipica configurazione di un sistema SCADA, [1]

# Capitolo 3

# Sicurezza nei sistemi ICS e protocolli industriali

Gli ICS originariamente erano dei sistemi isolati e la sicurezza era apparentemente garantita perché limitata alla rete del processo. Inoltre, i protocolli erano proprietari e i documenti non erano resi pubblici quindi questo amplificava il falso senso di intoccabilità [3]-[4]. Solamente i produttori e gli attaccanti conoscevano le vulnerabilità ma nessuno delle due parti aveva interesse nella loro divulgazione.
Tuttavia, nel tempo si è reso necessario interconnettere i sistemi ICS con la rete aziendale e Internet. Questo, in aggiunta all'incremento dell'uso di protocolli *open* e documentati e l'uso di tecnologie ICT/IT (*Information and Communications Technology*) ha esposto i sistemi industriali a seri problemi di sicurezza [10]-[11]-[12].

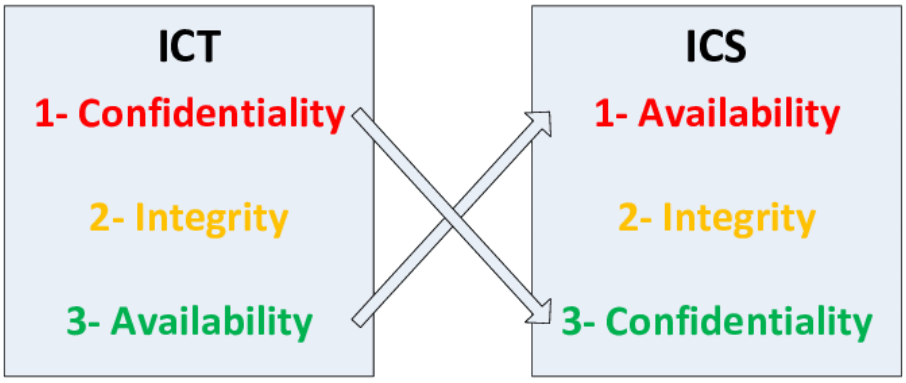


Figura 3.1: Priorità degli ICS e degli ICT a confronto

Come risultato le architetture SCADA stanno diventando sempre più simili ai sistemi ICT: se questo da una parte introduce significanti innovazioni e riduce drasticamente i costi, d' altra parte espone ancor di più gli ICS al mondo esterno [2].
Come mostra la figura 3.1 bisogna infatti considerare che le priorità di un sistema ICS sono diverse da quelle di un ICT [4]. Ad esempio nelle reti ICT la confidenzialità e la sicurezza hanno priorità massima, mentre risultano essere delle caratteristiche

marginali in ambito ICS. D' altro canto come anticipato nel capitolo 1, un sistema ICS per sua natura deve rispettare il requisito di *high-availability*, cosa che passa in secondo piano nell' ICT.
Ci sono anche altri caratteristiche dell'ambito ICS che rendono ancora più vulnerabile l' ambiente: la maturità della tecnologia e delle piattaforme è valutata come un riconoscimento implicito del valore e dell'affidabilità, oltretutto aggiornamenti continui rallenterebbero i processi facendo perdere ingenti somme alle aziende.
Anche le motivazioni degli attaccanti sono diverse quando si parla di ICT; mentre in questo caso lo scopo è quello del guadagno tramite riscatto, l' attacco ad un ICS viene fatto solitamente per mettere in ginocchio intere aree geografiche durante, ad esempio, le ***cyber war*** [13]-[14].

### 3.0.1 STUXNET

Il caso considerato da tutti il primo vero, grande attacco cibernetico della storia risale all'anno 2010 e porta il nome di **Stuxnet**, uno dei virus informatici più distruttivi mai realizzati.
Nel gennaio 2010, nella centrale di Natanz in Iran, le centrifughe dedicate all'arricchimento dell'uranio[1] impazzirono, andando fuori controllo: da 1.064 giri/minuto passarono ad una rotazione a 1.410 giri/minuto.
Stuxnet era in grado di agire sui **PLC Siemens Simatic S7-300**, adibiti al controllo delle centrifughe e tra le altre cose sfruttava ben quattro ***exploit*** cosiddetti ***"zero-day"*** [15]-[12]-[16].
Il *warm* infettò circa 200.000 computer e causò il degrado fisico di 1.000 macchine [17].

### 3.0.2 Statistiche recenti sugli attacchi ai sistemi industriali

Secondo il report ICS CERT di **Kaspersky**, in Italia, nel primo semestre del 2023, sono stati rilevati e bloccati oggetti malevoli sul 23,7% dei computer ICS.
Nella prima metà dell'anno, le soluzioni di sicurezza di Kaspersky hanno bloccato diverse famiglie di **malware** all'interno dei sistemi industriali. Tra queste, le più diffuse in Italia sono risultate script dannosi e pagine di phishing (9,7%), risorse Internet non segnalate (7,7%) e documenti malevoli (4,5%) (figura 3.2).

[1] L'uranio arricchito è un componente che può essere utilizzato per le armi nucleari, ed è molto spesso indispensabile per produzione di energia nucleare.

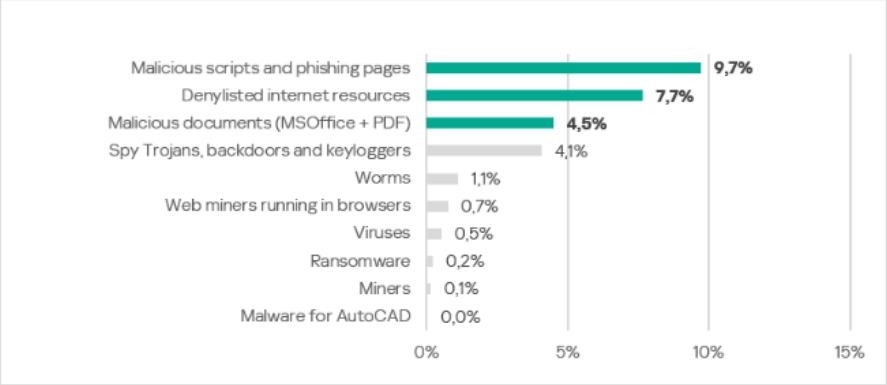


Figura 3.2: Percentuale di computer ICS su cui sono stati bloccati malware di diverse categorie in Italia nell'anno 2023.

Sempre nel primo semestre dell'anno, i paesi di solito più sicuri hanno subito cambiamenti inaspettati. Durante questo periodo, Australia, Nuova Zelanda, Stati Uniti, Canada, Europa occidentale e settentrionale, tipicamente caratterizzati da bassi livelli di minacce, hanno registrato un aumento nel numero di computer ICS attaccati. Per quanto riguarda l'Europa occidentale, il settore manifatturiero risulta essere il più colpito (17,4%), seguito da quello dell'energia (16,2%) ed oil&gas (12,2%) [18]. Si rende quindi necessario prevenire attacchi che potrebbero nuocere anche alla quotidianità del singolo individuo: si pensi ad un'industria che smette di fornire energia elettrica per giorni interi.

Purtroppo il problema della sicurezza in ambito ICS è stato ignorato per parecchi anni persistendo fino ad oggi, ad esempio vengono ancora ampiamente utilizzati protocolli non sicuri come **Modbus**.

In definitiva, applicare soltanto le soluzioni per la sicurezza degli ICT non risolve la questione in ambito ICS, perciò è importante trovare un approccio specifico per rilevare le minacce [13]-[14]-[8].

## 3.1 Protocolli industriali

Alla base dell'interconnessione tra un macchinario e un software c'è necessariamente un protocollo di comunicazione.
Capita spesso di vedere all'interno di un'azienda macchinari di età, fornitori e tecnologie differenti. Tale eterogeneità richiede spesso l'utilizzo di protocolli di comunicazione diversi rendendo quindi l'operazione di interconnessione complessa e non immediata.
In seguito, una breve carrellata dei principali protocolli di monitoraggio esistenti [19], con una più dettagliata descrizione di Modbus .

### 3.1.1 Ethernet/IP

È uno standard industriale *open* che nasce nel 2000 con l'obiettivo di avere un protocollo su Ethernet in grado di rispondere alle esigenze di performance ed affidabilità richieste dal mondo industriale. È un protocollo completo perché supporta il trasferimento di dati I/O basilari, il monitoraggio ciclico e su eventi di cambio di stato [20].
L'accesso alle variabili avviene per nome (TAG).
EtherNet/IP è un protocollo real-time con prestazioni discrete relativamente al suo dominio, che comunque si adattano perfettamente al campo del monitoraggio.
È un protocollo ben consolidato nel mercato americano, un po' meno in quello europeo. Viene spesso utilizzato con i sistemi di controllo Rockwell (PLC Allen Bradley) [21].

### 3.1.2 Profinet IO

Profinet utilizza Ethernet tradizionali per definire una rete con il compito di scambiare dati, allarmi e diagnostica con PLC e altri dispositivi industriali.
PROFINET IO utilizza tre diversi canali di comunicazione per lo scambio di dati con controllori programmabili e altri dispositivi. Il canale standard TCP/IP viene utilizzato per parametrizzazione, configurazione e operazioni di lettura/scrittura acicliche. Il canale real-time viene utilizzato per il trasferimento di dati ciclici standard e gli allarmi. Il terzo canale, *Isochronous Real Time* (IRT) è il canale ad altissima velocità utilizzato per le applicazioni *Motion Control*[2]. Essendo un protocollo *real-time*, presenta prestazioni elevate che lo rendono adatto al controllo veloce. I requisiti hardware di Profinet sono spesso restrittivi quando si ha la necessità di utilizzare un dispositivo non real-time. Una necessità frequente nel campo del monitoraggio è quella di collegarsi da un PC con sistema operativo Windows verso il campo. In questi casi, spesso, si preferisce appoggiarsi ad un altro protocollo (magari anche con prestazioni inferiori).
Profinet è utilizzato dai PLC e dispositivi del mondo Siemens [22]-[23].

### 3.1.3 MTCONNECT

È uno standard industriale utilizzato per la raccolta dati e il monitoraggio sulle macchine a controllo numerico (CNC). Il protocollo supporta solo la lettura dei dati, che sono presentati in formato XML.
La prima dimostrazione pubblica di MTConnect è avvenuta all'International Manufacturing Technology Show (IMTS) tenutosi a Chicago nel settembre 2008. Lì, 25 produttori di apparecchiature industriali collegarono in rete i loro sistemi di controllo delle macchine, fornendo informazioni sui processi che potevano essere recuperati

[2]Il motion controller è uno speciale dispositivo che controlla la modalità di funzionamento del motore. Ha il compito di dire al motore cosa fare in base alla lavorazione da eseguire.

da qualsiasi client abilitato al web connesso alla rete.
MTConnect è spesso disponibile su CNC Mazak, Mitsubishi e Mori Seki [24].

### 3.1.4 OPC-UA

È un protocollo industriale multipiattaforma e aperto, sviluppato dalla OPC Foundation. UA sta per *Unified Architecture*.
OPC UA soppianta il vecchio OPC, mantenendo tutte le funzionalità del suo predecessore. La forza di OPC-UA è senza dubbio la sua flessibilità. Oltre ai PLC dell'ambito industriale, è infatti supportato da Windows, Linux, iOS ed anche Android [25].

### 3.1.5 Modbus

È il più longevo tra i protocolli presentati. Nasce infatti nel 1979 dalla Schneider Automation [26] e diventa in breve tempo uno standard de facto.
Nonostante ci siano protocolli alternativi che presentano prestazioni migliori, il protocollo Modbus è largamente utilizzato perché è *open source*, ha un basso costo di sviluppo e richiede hardware limitato. Non presenta velocità elevate ma è perfettamente adatto per il monitoraggio dove le velocità richieste sono, appunto, contenute. Modbus è spesso utilizzato dai dispositivi Schneider ma anche su dispositivi più semplici come sensori di temperatura e per il monitoraggio energetico.
È un protocollo di messaggistica a livello applicativo (livello 7) del modello OSI.
Nel protocollo Modbus, ogni slave memorizza le informazioni in quattro differenti tabelle da 9.999 valori ciascuno. Due tabelle mantengono valori booleani (*coils* o *contacts*) e le altre due valori numerici in parole da 16 bit (registri):

- *Discrete Output Coils*, per operazioni di lettura e scrittura su registri booleani. Questo tipo di tabella potrebbe essere utile per leggere o impostare lo stato LED acceso/spento in un dispositivo.
- *Discrete Input Contacts*, per operazioni di sola lettura su registri booleani. Possono essere sfruttati per leggere lo stato di un sensore che si attiva in seguito ad un *trigger* esterno.
- *Analog Input Registers*, per operazione di sola lettura su registri numerici. Ad esempio potrebbero contenere informazioni sulla temperatura.
- *Analog Output Holding Registers*, per operazione di lettura e scrittura su registri numerici. Utili ad esempio per contenere/modificare i dati di configurazione di un dispositivo.

I valori sono letti o scritti facendo riferimento agli indirizzi dati delle rispettive tabelle, ossia valori esadecimali compresi tra 0000h e 270Eh (9998 in decimale), con relativo offset escluso (figura 3.3).

| Indirizzi dati | Offset | Numero consecutivo associato all'elemento analogico o discreto | Tipo | Nome tabella |
|---|---|---|---|---|
| 0000h - 270Eh | 00001 | 00001-09999 | R/W | Discrete Output Coils |
| 0000h - 270Eh | 10001 | 10001-19999 | Read-Only | Discrete Input Contacts |
| 0000h - 270Eh | 30001 | 30001-39999 | Read-Only | Analog Input Registers |
| 0000h - 270Eh | 40001 | 40001-49999 | R/W | Analog Output Holding Registers |

Figura 3.3: Organizzazione dei registri in tabelle

Ogni produttore di dispositivi Modbus deve dotare il proprio prodotto di un mezzo per rimappare i valori impiegati nelle effettive locazioni di memoria utilizzate per memorizzare i dati di funzionamento. In questo modo si crea un livello di astrazione che permette agli utilizzatori Modbus di accedere ai parametri di funzionamento del dispositivo in maniera trasparente e indipendente dai dettagli della particolare implementazione [27].

### Richiesta

I messaggi Modbus che vengono inviati dal master allo slave contengono:

- L' indirizzo dello slave destinatario.
- Il campo *function code* che contiene il codice dell' operazione che lo slave deve eseguire (ad esempio "leggi registro", "scrivi registro").
  Esistono anche operazioni di diagnostica che permettono di leggere informazioni aggiuntive riguardo le caratteristiche dello slave e device (e.g., operazione 43 *"Read device Identification"* e l'operazione 17 *"Report Slave ID"* per la modalità seriale) [28].
  Oltre alle funzioni pubbliche ci sono codici riservati ai produttori per personalizzare l'implementazione [29].
  Alcune delle associazioni codice-azione pubbliche sono elencate nella figura 3.4.
- I byte di dati contengono ulteriori informazioni per elaborare la richiesta: ad esempio parametri addizionali che indicano punto di partenza e fine di una lettura di registri.
- La *check-sum* fornisce un metodo allo slave per validare l'integrità del contenuto del messaggio [3].

[3] I metodi utilizzati sono l' LRC - *Longitudinal redundancy check* oppure il CRC - *Cyclic redundancy check.*

| Codice funzione | Azione | Effettuata sulla tabella |
|---|---|---|
| 01 (01h) | Read | Discrete Output Coils |
| 02 (02h) | Read | Discrete Input Contacts |
| 03 (03h) | Read | Analog Output Holding Registers |
| 04 (04h) | Read | Analog Input Registers |
| 05 (05h) | Write single | Discrete Output Coils |
| 15 (0Fh) | Write multiple | Discrete Output Coils |
| 06 (06h) | Write single | Analog Output Holding Registers |
| 16 (10h) | Write multiple | Analog Output Holding Registers |

Figura 3.4: Codici funzione [27]

### Risposta

Nel caso in cui non si siano verificati errori, lo slave genera un messaggio di risposta in cui viene ricopiato il codice della funzione dal messaggio di richiesta. I byte di dati conterranno informazioni collezionati dallo slave, come valori di registri o lo stato. Se, viceversa, dovesse occorrere un errore il codice della funzione viene modificato per indicare che la risposta è un messaggio d'errore e i byte di dati vengono usati per descriverlo. Come prima, l' *error check field* permette al master di stabilire se il contenuto del messaggio è valido.

### Broadcast

L'indirizzo 0 è tipicamente riservato per il *broadcast*.
Quando il master genera un messaggio con destinatario 0 questo viene recapitato a tutti gli slave da esso controllati. Possono essere richieste operazioni di *write* che prevedono che i destinatari settino i propri registri/*coils* con i valori previsti. Per quanto riguarda le operazioni di lettura, invece, nel caso di collegamenti seriali non vengono generate risposte poiché potrebbero verificarsi conflitti nell'uso dei bus fisici [30]. Diverso è il caso Modbus TCP dove la configurazione dello slave 0 è a discrezione dei produttori.
In seguito verranno descritte tre versioni di Modbus che utilizzano diversi protocolli di comunicazione dati seriali, quindi nel livello 1 (fisico) del modello OSI. Il Modbus TCP viene implementato sfruttando Ethernet mentre per Modbus ASCII e Modbus RTU vengono tipicamente usati i protocolli RS232, RS422 o RS485 [31]-[32]-[26]. Proprio questa differenza rende la prima versione preferibile alle altre due in termini di velocità, motivo per cui il Modbus TCP sarà trattato in una sezione a parte.

### Modbus ASCII

Quando i controllori sono impostati per comunicare su una rete Modbus in questa modalità, ogni byte viene inviato come due caratteri ASCII (*American Standard Code for Information Interchange*). Il vantaggio è che questo consente di avere intervalli di tempo fino a un secondo tra un carattere e l'altro senza causare errori.

### Modbus RTU

Quando i controllori sono impostati per comunicare su una rete Modbus utilizzando la modalità RTU (*Remote Terminal Unit*), ogni byte di un messaggio contiene due caratteri esadecimali a quattro bit e deve essere trasmesso in un flusso continuo. Il vantaggio principale di questa modalità è che la maggiore densità di caratteri consente una migliore velocità di trasmissione dei dati rispetto all'ASCII. Quando si usa questa versione del protocollo gli slave possono ricevere comandi da un solo master (3.5) [32].

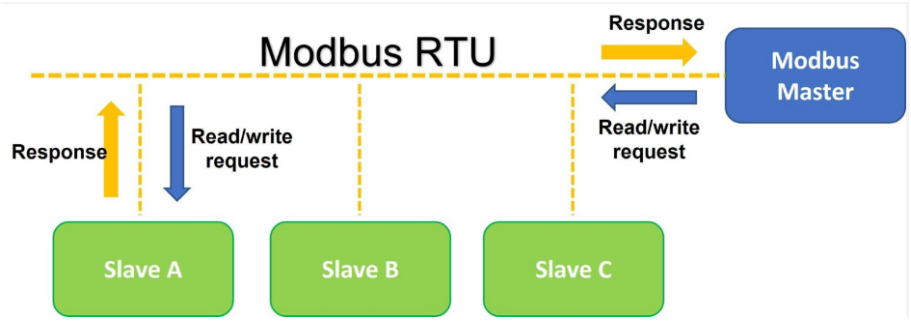


Figura 3.5: Modbus RTU [33]

## 3.2 Modbus TCP

Il principio su cui si basa questa versione di Modbus è simile alla modalità RTU con la differenza che i dati vengono trasmessi dentro pacchetti TCP/IP: questo consente a più master di collegarsi allo stesso slave a patto di utilizzare una diversa porta TCP. Altra differenza è che Il campo di controllo degli errori utilizzato in Modbus RTU viene eliminato poiché il livello di collegamento TCP/IP utilizza i suoi metodi di *check-sum* [34].
Il fatto di sfruttare il protocollo TCP/IP fa sì che Modbus possa comunicare su reti Ethernet, rendendo la trasmissione di dati più veloce rispetto alla modalità RTU (figura 3.6) [33]-[35].

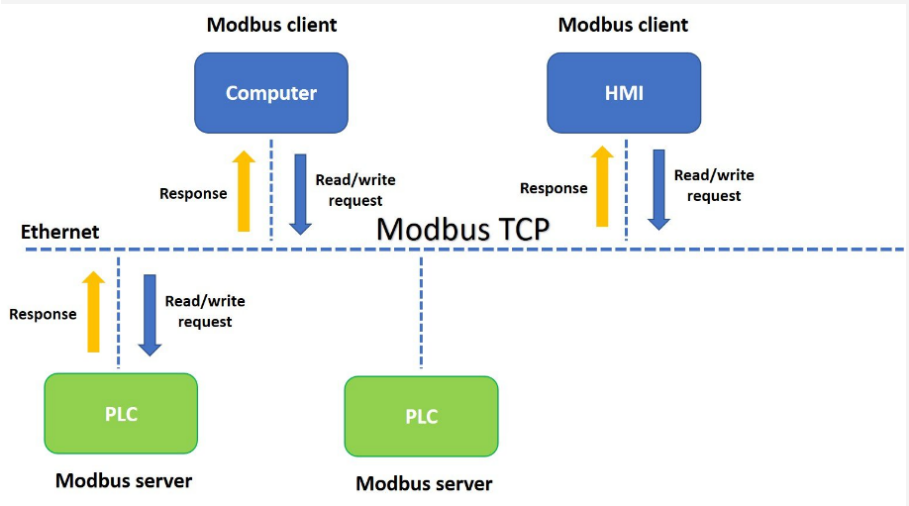


Figura 3.6: Modbus TCP [33]

Un sistema di comunicazione basato su Modbus TCP/IP potrebbe includere diversi tipi di dispositivi:

- Modbus TCP/IP client e server connessi a una rete TCP/IP
- Device tipo bridge, router o gateway per l' interconnessione tra reti TCP/IP e sotto-reti seriali.

Il protocollo Modbus definisce una semplice unità di dati chiamata **PDU** (*Protocol Data Unit*) che è indipendente dai livelli sottostanti. Il client che avvia una transazione Modbus costruisce l' **ADU** (*Application Data Unit*) nella quale si differenziano diversi campi: alcuni facoltativi ed altri, come la *function code*, obbligatori (figura 3.7) [32].

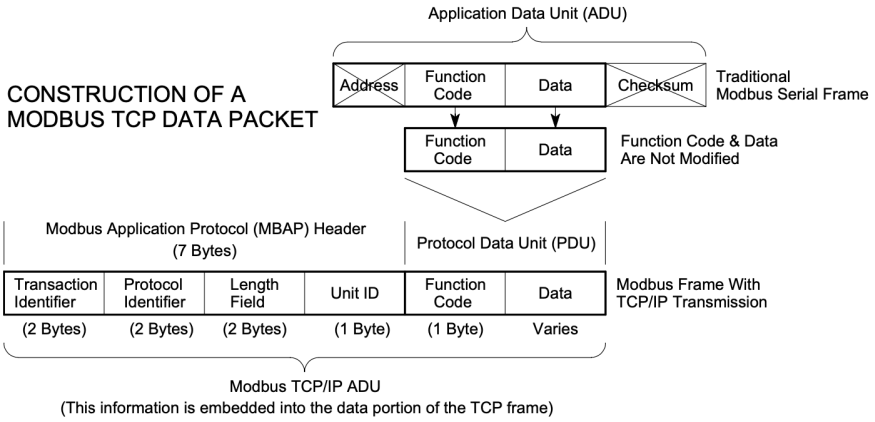


Figura 3.7: Pacchetti Modbus TCP

### 3.2.1 ADU Modbus su TCP/IP

Come visibile nella figura 3.8 l' incapsulamento di una richiesta o di una risposta nel caso del Modbus TCP richiede un campo ulteriore a quelli presentati per il Modbus tradizionale: l' **MBAP** (*Modbus Application Protocol*). Quest'intestazione prevede delle differenze rispetto al Modbus RTU:

- Nell'MBAP viene utilizzato un solo byte per l' indirizzo dello slave. Inoltre, in caso di comunicazione via dispositivi come bridge, router e gateway, l'MBAP contiene un sotto-campo chiamato *unit ID*. Questi device usano un singolo indirizzo IP per supportare molteplici unità terminali Modbus indipendenti.
- Tutte le richieste e le risposte Modbus sono progettate in modo che il destinatario possa verificare che la trasmissione del messaggio sia conclusa. Per le funzioni per cui il Modbus PDU non ha una lunghezza prefissata, e che quindi trasportano dati di lunghezza variabile nella richiesta o nella risposta, è necessario che nel campo *data* venga mantenuto un contatore di byte.
- In caso di frammentazione dei pacchetti le informazioni addizionali relative alla lunghezza vengono inserite nel MBAP, di modo che il destinatario possa ricostruire in maniera esatta il messaggio.

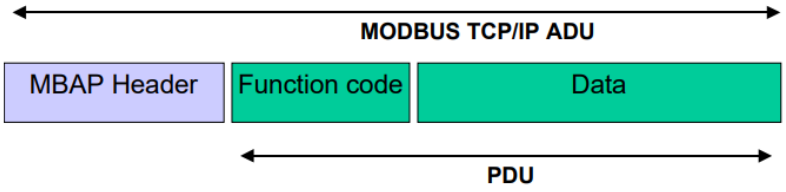


Figura 3.8: Richiesta/Risposta Modbus TCP

### Descrizione dei campi dell'MBAP

Nella figura 3.9 vengono descritti brevemente i campi dell' MBAP

| Campi | Lunghezza | Descrizione | Client | Server |
|---|---|---|---|---|
| Transaction Identifier | 2 Bytes | Identifica richiesta\risposta ModBus | Inizializzato dal Client | Ricopiato dal Server dalla richiesta ricevuta |
| Protocol Identifier | 2 Bytes | 0 = protocollo ModBus | Inizializzato dal Client | Ricopiato dal Server dalla richiesta ricevuta |
| Lunghezza | 2 Bytes | Numero di byte | Inizializzato dal Client nella richiesta | Inizializzato dal Server nella risposta |
| Unit Identifier | 1 Bytes | Identificatore di uno *slave* remoto su una linea seriale o su altri bus | Inizializzato dal Client | Ricopiato dal Server dalla richiesta ricevuta |

Figura 3.9: MBAP

## 3.2.2 Vulnerabilità del protocollo Modbus TCP

Come viene scritto nel blog di Omar Morando [34] l'implementazione del protocollo Modbus TCP contiene diverse vulnerabilità che potrebbero consentire a un utente malintenzionato di eseguire attività di enumerazione o di inviare comandi arbitrari. I punti chiave sono:

- Mancanza di riservatezza: tutti i messaggi Modbus vengono trasmessi in chiaro attraverso il supporto di trasmissione.

- Mancanza di integrità: non esistono controlli di integrità all'interno del protocollo e, di conseguenza, dipende da protocolli di livello inferiore preservare l'integrità dei dati [12].

- Mancanza di autenticazione: non esiste autenticazione a nessun livello del protocollo, con la possibile eccezione di alcuni comandi di programmazione non documentati.

- Mancanza di struttura della sessione: non esiste un concetto di sessione che persiste tra più richieste o transazioni. Come molti protocolli di richiesta/risposta (es. SNMP, HTTP, ecc.) Modbus TCP è costituito da transazioni di breve durata in cui il master invia una richiesta allo slave che si traduce in una singola azione. Se combinato con la mancanza di autenticazione e la scarsa generazione del TCP Initial Sequence Number (ISN) in molti dispositivi embedded, diventa possibile per gli aggressori immettere comandi senza conoscere la sessione esistente.

- Framing semplicistico: i frame Modbus TCP vengono inviati tramite connessioni TCP stabilite. Sebbene tali connessioni siano generalmente affidabili, presentano un significativo svantaggio per via del punto precedente.

### Vulnerabilità "Illegal Function Exception"

Queste vulnerabilità consentono a un utente malintenzionato di svolgere attività di ricognizione sulla rete di destinazione. La prima vulnerabilità esiste perché un dispositivo slave Modbus può restituire una *Illegal Function Exception* per le *query* che contengono un codice funzione non supportato. Un utente remoto non autenticato può sfruttare questa vulnerabilità inviando codici funzione predisposti per effettuare ricognizioni sulla rete di destinazione.

### Vulnerabilità "Illegal Address Exception"

Un'ulteriore vulnerabilità di ricognizione è dovuta alle molteplici risposte di *Illegal Address Exception* generate per le *query* che contengono un indirizzo slave illegale. Un utente malintenzionato non autenticato può sfruttare questa vulnerabilità inviando query che contengono indirizzi non validi alla rete di destinazione e raccogliere informazioni sugli host di rete dai messaggi restituiti.

### Vulnerabilità sull'autenticazione

Un'altra vulnerabilità è dovuta alla mancanza di controlli di sicurezza nell'implementazione del protocollo Modbus TCP. Le specifiche del protocollo non includono un meccanismo di autenticazione per la convalida della comunicazione tra i dispositivi master e slave. Questo difetto potrebbe consentire a un utente non autenticato di inviare comandi arbitrari a qualsiasi dispositivo slave tramite un master di attacco.

### Vulnerabilità DoS

Il protocollo Modbus TCP contiene anche vulnerabilità che potrebbero consentire a un utente malintenzionato di causare una condizione di *denial-of-service* (DoS) su un sistema di destinazione. La vulnerabilità è dovuta a un errore di implementazione nel protocollo stesso durante l'elaborazione dei messaggi di richiesta e risposta di input discreti.

### Vulnerabilità di buffer overflow

Un altro attacco a Modbus può essere il pacchetto dati che supera la lunghezza massima. Il protocollo limita la dimensione della PDU a 253 byte per consentire l'invio del pacchetto su una linea seriale, es. interfaccia RS-485.
Modbus TCP antepone alla PDU un'intestazione *Modbus Application Protocol* (MBAP) di 7 byte e il tutto, MBAP+PDU, viene incapsulato in un pacchetto TCP. Ciò pone un limite massimo alla dimensione del pacchetto.

Un utente malintenzionato crea un pacchetto appositamente predisposto di lunghezza superiore a 260 byte e lo invia a un client e server. Se il client o server sono stati programmati in modo errato ciò potrebbe provocare un *overflow* del buffer o un attacco *denial-of-service*.

### Sniffing del protocollo

L'attacco più semplice da usare contro Modbus è lo sniffing del traffico di rete, trovare i dispositivi connessi e quindi inviare comandi dannosi ai dispositivi.
Non avendo funzionalità di sicurezza o crittografia, è facile utilizzare *Wireshark* per raccogliere informazioni da pacchetti di dati da e verso una porta Modbus su un dispositivo. *Wireshark* consente di leggere il contenuto di tali pacchetti, esaminare gli indirizzi IP, vedere i codici funzione delle richieste e alterare il corretto funzionamento dei dispositivi.

# Capitolo 4

# Generalità degli Honeypot

Per definizione, un ***honeypot*** è un'esca o un obiettivo fittizio nato nel settore dell' IoT (*Internet of Things*) [14]-[8] creato per attirare, rilevare e osservare il comportamento degli intrusi. Motivo per cui viene esposto di proposito a *probing* e ad altri tipi di attacchi. L'implementazione e la gestione di un'infrastruttura honeypot richiede perciò molta attenzione: l' honeypot deve risultare trasparente all' attaccante affinché non possa essere sfruttato a favore di quest'ultimo per aumentare la superficie di attacco.
Questo strumento non esclude l' utilizzo di *firewall* e antivirus, anzi, se affiancati rafforzano la sicurezza di un sistema [36].
Poiché gli honeypot non hanno alcun scopo in una rete se non quello di ingannare utenti malevoli, il tipo di interazione con esso può essere determinante per comprendere se l' attacco è indiscriminato oppure è lacunoso.
Nel primo caso l'attacante cerca un qualsiasi dispositivo vulnerabile quindi potrebbe continuare a interagire con l'honeypot per individuarne gli apparenti punti deboli.
Nella seconda ipotesi l'attaccante ha meno conoscenze ma ha comunque un obiettivo specifico: potrebbe dover eseguire una scansione di una rete per individuare il dispositivo di destinazione. In tal caso, interagirà con l'honeypot almeno una volta e lascerà delle tracce del tentato attacco.
Viceversa se un attaccante ha conoscenze sufficienti e un obiettivo specifico, può interagire direttamente con il dispositivo di destinazione in una rete e lasciare intatti gli honeypot, rendendo quindi vano il loro utilizzo [14].

Gli honeypot possono essere classificati in due gruppi:

- **honeypot di ricerca**: utilizzati per ottenere informazioni riguardo i metodi di attacco. Sono utili per rilevare e monitorare attacchi indiscriminati su larga scala, ma non attacchi mirati e informati [14].

- **honeypot di produzione**: proteggono implicitamente l' infrastruttura fornendo un preavviso in caso di attacchi.

Sempre per quanto riguarda i tipi di honeypot, esiste anche un'altra distinzione [37] che può essere stabilita tra **honeypot server** e **honeypot client**. Il primo tipo è progettato per attendere passivamente gli attacchi, mentre il secondo è in grado di cercare attivamente i server dannosi e di comportarsi come una vittima [13].
I tipi di honeypot possono anche essere distinti in base alla capacità dell'attaccante di interagire con l'applicazione o i servizi [38]:

- Gli **honeypot ad alta interazione** sono dotati di una logica interna e possono essere sondati, attaccati e compromessi. L' obiettivo è quello di catturare quante più informazioni possibili riguardo le tecniche di intrusione e vedere quali sono i punti deboli del sistema che vengono sfruttati. Questo tipo di honeypot permette agli attaccanti di operare in libertà una volta che il sistema è stato compromesso, quindi richiede un attento monitoraggio e un' analisi dettagliata.

- Gli **honeypot a bassa interazione** emulano le vulnerabilità e vengono usati spesso come esche. Rispetto agli honeypot ad alta interazione risultano essere più sicuri perché viene limitata la capacità di interazione dell' attaccante con il sistema, d'altro canto però sono meno flessibili [39]. **Conpot**, che verrà approfondito in seguito, fa parte di questa categoria.

Gli honeypot a bassa interazione sono economici da realizzare ma facili da identificare. Gli honeypot ad alta interazione sono molto costosi ma meno limitati. Un'alternativa potrebbe essere quella di realizzare un dispositivo ibrido che sia a bassa interazione ma che si comporti anche come un *proxy*. Quando si comporta come proxy l' honeypot redirige il traffico verso un device di produzione (e.g., PLC, RTU) e si comporta come un *man-in-the-middle* tra la rete e il dispositivo [14].
Esistono anche honeypot a **media interazione**: sistemi che hanno le stesse funzionalità degli honeypot a bassa interazione ma che in più sono in grado di simulare un sistema semplice.

## 4.1 Honeypot

Esistono diversi honeypot in circolazione che si differenziano tra loro per alcune caratteristiche. Pertanto nelle prossime sezioni verranno analizzate sinteticamente alcune delle loro peculiarità.

### 4.1.1 HoneyD

HoneyD è un honeypot a bassa interazione lato server che emula diversi servizi e sistemi operativi. Lo scopo principale di Honeyd è il rilevamento di attività non autorizzate all'interno di una rete. Questo avviene attraverso il monitoraggio degli indirizzi IP non utilizzati di uno specifico intervallo. Ogni connessione verso uno di tali indirizzi viene considerato come un possibile attacco o comunque come il

risultato di una scansione. Permette la creazione di un ambiente di rete su di una singola macchina fisica. Honeyd fornisce meccanismi per il rilevamento e la valutazione delle minacce, inoltre scoraggia gli avversari camuffando sistemi reali in mezzo a sistemi virtuali. Supporta nativamente i protocolli TCP, UDP (*User Datagram Protocol*) e ICMP (*Internet Control Message Protocol* ) [39] [40] [41].

### 4.1.2 Capture-HPC

Capture-HPC è un' applicazione che cerca attivamente server malevoli. In particolare quelli che attaccano *web browser* o qualsiasi altra applicazione che accede a materiale remoto o al sistema operativo. Essendo un sistema lato client e ad alta interazione, si presenta al server remoto potenzialmente dannoso come un sistema operativo con un set di applicazioni autentiche sconosciute all'aggressore. Tutti questi strumenti esistono all'interno di un ambiente virtuale che comprende tool di registrazione e di analisi che rilevano attività dannose [42].

### 4.1.3 HoneyPLC

HoneyPLC è un honeypot a media interazione specifico per i sistemi ICS. Questo sistema è capace di simulare in maniera fedele i controllori logici programmabili provenienti da vari fornitori, è estensibile e raccoglie informazioni riguardo *malware* per sistemi industriali. Può registrare interazioni S7comm[1], supporta SNMP (*Simple Network Management Protocol*, utilizzato per la gestione della rete) e HTTP (*Hypertext Transfer Protocol*) [43].

### 4.1.4 LOGistICS

LOGistICS, è un sistema di emulazione e monitoraggio per ICS sviluppato durante un lavoro di tesi magistrale. Il cuore di LOGistICS è rappresentato da un honeypot per ICS. I protocolli supportati sono S7comm e Modbus. LOGistICS è composto da:

- un honeypot altamente personalizzabile che emula i due suddetti servizi ICS
- uno sniffer che registra il traffico
- un nodo di monitoraggio, in grado di analizzare e tracciare i dati registrati

Inoltre è stato implementato un ulteriore nodo di test localmente distribuito con lo scopo di valutare la capacità dell'honeypot nel registrare efficacemente tali eventi. I modelli di PLC emulati sono rappresentati attraverso file csv, ed è quindi possibile aggiungerne nuovi a piacimento [8].

[1] Protocollo proprietario strettamente associato a Siemens

### 4.1.5 GasPot

GasPot è un honeypot progettato da Veeder Root per simulare un sistema di monitoraggio dei serbatoi chiamato Guardian AST. Questi sistemi di monitoraggio sono comunemente usati nell'industria petrolifera per misurare il livello di carburante nei serbatoi [44].

### 4.1.6 Gridpot

Gridpot è un honeypot a media interazione Conpot-based che simula i dispositivi di controllo industriale, la loro rete elettrica e il flusso di elettricità attraverso la rete. Comprende due componenti principali: un simulatore di rete elettrica chiamato GridLAB-D e un honeypot basato su Conpot [45].

### 4.1.7 Conpot

Conpot, come accennato nei paragrafi precedenti, è un honeypot a bassa interazione. Il suo obiettivo è quindi, quello di raccogliere informazioni riguardo i motivi e i metodi usati da un avversario che prende di mira un ICS. Supporta la simulazione di protocolli HTTP, SNMP e Modbus. Permette inoltre l'integrazione di controllori logici programmabili (PLC).
L'honeypot registra gli eventi dei servizi HTTP, SNMP e Modbus con una precisione al millisecondo e offre informazioni di base sul tracciamento, come l'indirizzo di origine, il tipo di richiesta e la risorsa richiesta nel caso di HTTP [46]. Sviluppato dalla **MushMushFoundation**, è reperibile online [47].
Conpot ha una struttura modulare che comprende i seguenti componenti:

- *Protocol Server*: questi moduli riconoscono e a volte rispondono al traffico usando uno specifico protocollo del livello applicativo.
- *Core*: durante l'avvio questo componente legge il file di configurazione principale per determinare quali *protocol server* creare e quali porte assegnare a ciascun server. Inizializza anche un *bus* di dati virtuali che permette ai server di scrivere e leggere dati senza generare effettivamente attività di rete. Quindi, se viene scelto di usare il protocollo Modbus il *core* inizializza il server appropriato (server Modbus in questo caso). Successivamente ascolta il traffico in entrata e lo reindirizza alla porta del server opportuno. Il *core* assegna inoltre a ogni combinazione di IP remoto e porte TCP/UDP un identificatore (ID) univoco composto da 32 cifre esadecimali. L'ID viene aggiunto a tutte le voci del registro eventi associate ai *socket* remoti opportuni.
- *Loggers*: sono dei processi indipendenti che ricevono notifiche dagli eventi dal *protocol server* in esecuzione e generano un singolo file di log in diversi formati. Il *Conpot core* automaticamente tiene traccia degli eventi dall'inizio alla fine della sessione, ma una volta che il controllo viene passato al *protocol server* è lui stesso a decidere cos'altro deve essere registrato.

- *Templates*: ogni *protocol server* ha un *template* che contiene le informazioni di setup per il server in questione. Di default Conpot simula un PLC Simatic S7-200 ma può essere esteso e modificato editando dei file XML.

## 4.2 Architettura di riferimento di un Honeypot Modbus in ambito SCADA

Solo di recente stanno iniziando a diffondersi anche tra i sistemi ICS mezzi come gli honeypot. Le infrastrutture dei sistemi SCADA sono solitamente composte da due livelli:

- la rete operativa, dove si trovano il server centrale (*master station*) e i database
- una o più *field network*[2] dove sono posizionati diversi PLC ed RTU distribuiti che controllano un processo critico

Oltre a questi due livelli, che sono specifici per il funzionamento dello SCADA, c'è anche la rete IT dell'organizzazione che comprende altri dispositivi (*workstation*, server) e servizi.
L'honeypot Modbus presentato da Simoes, Cruz, Gomez e Monteiro (figura 4.1) [13] è progettato per operare nell' ambito delle reti di un sistema SCADA/ICS, coesistendo con gli array di PLC esistenti, RTU e sensori/attuatori che popolano la rete. L' obiettivo principale è quello di emulare fedelmente il comportamento e il servizio di un PLC commerciale.

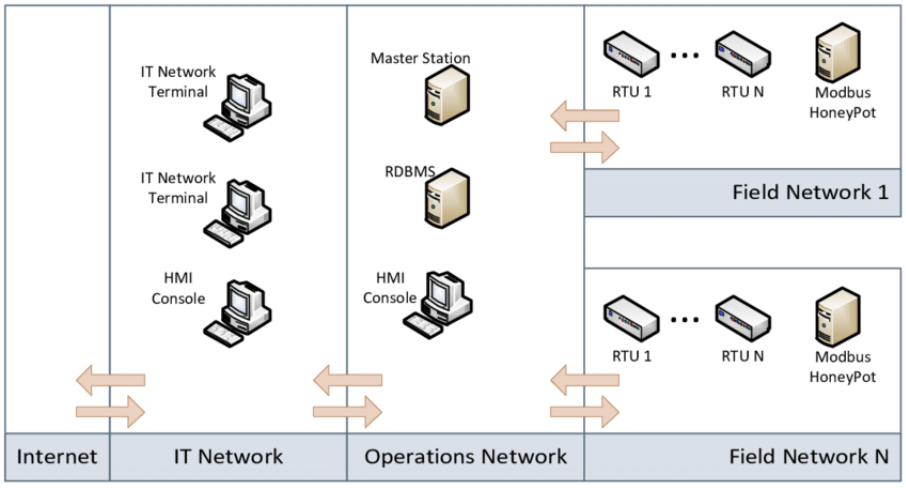


Figura 4.1: Posizione di un honeypot Modbus in un ambiente ICS

[2]Mezzo per ottenere produzioni e lavorazioni fluide tramite lo scambio di informazioni usando controllori (PLC), dispositivi I/O (sensori e indicatori), dispositivi di controllo e dispositivi di comunicazione (PC).

L' honeypot riporta le attività sospette, mandando eventi agli IDS [3] distribuiti degli ICS, dove poi saranno processati e associati. Di solito questo honeypot aspetta il tentativo di connessione da qualcuno che fa *probing* sulla rete o che sta tentando di accedere con l' intento di impersonare il server centrale.
Ogni tentativo di contattare l' honeypot potrebbe generare un allarme poiché, per definizione, ogni attività nell' honeypot Modbus (operazioni di gestione a parte) è illegale e non autorizzata. In poche parole un utente benigno non avrebbe motivo di contattare un honeypot.
Tuttavia gli honeypot presentano comunque un numero basso di falsi positivi [48][4].

## 4.3 Honeypot a confronto

Nello scenario ICS non c'è un' ampia scelta di honeypot, specialmente se siamo alla ricerca di sistemi a basso livello di interazione per la mera raccolta di dati. In seguito un grafico (figura 4.2) che mette a confronto alcune delle caratteristiche dei sistemi brevemente descritti sopra.

| | HoneyD | Capture-HPC | Honey PLC | LOGistICS | Conpot | GasPot | Gridpot |
|---|---|---|---|---|---|---|---|
| Versione Windows | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | X |
| Versione Linux | ✓ | ✓ | X | ✓ | ✓ | ✓ | ✓ |
| Lvl interazione | bassa | alta | media | media | bassa | bassa | media |
| Protocollo Modbus | X | X | X | ✓ | ✓ | X | ✓ |
| Open Source | ✓ | ✓ | ✓ | X | ✓ | ✓ | ✓ |
| Estensibilità | ✓ | X | ✓ | csv | xml | X | ✓ |

Figura 4.2: Honeypot a confronto

### 4.3.1 Considerazioni sulla scelta finale dell'honeypot Conpot

Nonostante gli honeypot ad alta interazione risultino essere più flessibili, in un primo approccio alla tematica dei suddetti è stato ritenuto consono utilizzare uno strumento più sicuro. Come è stato anticipato precedentemente infatti, il rischio di compromissione è ridotto poiché vengono emulati soltanto servizi di base. Inoltre, per tale motivo questi tipi di honeypot risultano essere anche più facili da configurare. Conpot in particolare utilizza un file XML di facile comprensione per la configurazione di un device, rendendo semplice, ma allo stesso tempo flessibile questa fase del processo.
Un'altra caratteristica specifica di Conpot è il fatto che sia stato implementato per simulare servizi e protocolli industriali, ne consegue la presenza nativa del protocollo Modbus. Altre alternative (e.g., Honeyd) avrebbero necessitato di tool aggiuntivi

[3] Intrusion Detection System
[4] In *cybersecurity* i falsi positivi sono degli allarmi che indicano erroneamente che si sta verificando un attacco. Falsi allarmi.

per integrarlo. Essendo lo scopo la simulazione di un semplice PLC con conseguente raccolta di dati quello che mette a disposizione Conpot è sufficiente, senza dover scomodare sistemi complessi e più pesanti computazionalmente parlando. Conpot è ampiamente utilizzato in ambito ICS e quindi sono presenti molte risorse in rete che riguardano questo specifico utilizzo. Il lato negativo di questo strumento è sicuramente la scarsa documentazione ufficiale. Molte sezioni utili, specialmente a chi si approccia a questo argomento per la prima volta, sono incomplete o inesistenti. La fase dell'installazione su host fa riferimento a pacchetti obsoleti e quindi ( almeno nel mio personale caso ) a ogni tentativo di installazione o all'avvio venivano generati errori che non permettevano la normale attività dell'honeypot. Per ovviare a questo problema si è ritenuto opportuno proseguire con l'installazione usando i *container* Docker. In definitiva, seppur Conpot abbia degli aspetti negativi è comunque quello che risulta essere più consono ai fini di questo studio.

# Capitolo 5

# Testing, raccolta dati e analisi utilizzando Conpot

In seguito verranno discusse le varie fasi eseguite utilizzando Conpot: dal *testing* al *deployment*.

## 5.1 Prerequisiti

### 5.1.1 Docker

I *container* Docker sono delle unità software che racchiudono un' applicazione e tutte le sue dipendenze. Questo permette di eseguire il software in maniera uniforme in diversi ambienti computazionali, rendendolo quindi portabile. In aggiunta, essendo i *container* isolati dall'hardware sottostante e dal sistema operativo, offrono un livello di sicurezza ulteriore. Il *container* è un'istanza dell'immagine. Un'immagine contiene tutte le istruzioni (e.g., sistema operativo, librerie, dipendenze) affinché un *container* possa essere eseguito.

## 5.2 Test preliminari

### 5.2.1 Installazione

Per l' installazione è stato scelto di utilizzare un *container* Docker, costruendo l' immagine dal codice sorgente:

- git clone `https://github.com/mushorg/conpot.git`
- sudo make

Passiamo alla creazione effettiva del container:

- docker create -it -p 80:8800 -p 102:10201 -p 502:5020 -p 161:16100/udp -p 47808:47808/udp -p 623:6230/udp -p 21:2121 -p 69:6969/udp -p 44818:44818 –network=bridge –name conpot conpot:latest

Durante la creazione del container con il flag -p vengono mappate le porte del container alle porte dell' host.
Per esempio, in questo caso la porta 5020 del container viene mappata alla 502 dell' host, usata tipicamente da Modbus.
Con l' opzione -network=bridge il *container* utilizza la rete bridge di default creata da Docker; viene mantenuto quindi l'isolamento dallo spazio di rete dell'host.

### 5.2.2 Personalizzazione

Come viene analizzato nella tesi *"Evasion of Honey- pot detection mechanisms through improved interactivity of ICS-based systems"* [45] Conpot è facilmente fruibile e questo ne ha semplificato il processo di *fingerprinting*: un avversario potrebbe quindi facilmente eludere o tentare di eliminare l' honeypot (ad esempio sfruttando attacchi Distributed Denial of Service (DDoS) [48]. Questo violerebbe i principi su cui si fonda l'honeypot rendendolo totalmente inutile [45].
La prima cosa più semplice da fare per rendere Conpot meno riconoscibile potrebbe essere quello di modificare il *template* di default, poiché è disponibile online.
I due file da modificare sono **modbus.xml** e **template.xml**.
Nel primo possono essere manipolate impostazioni riguardanti il PLC simulato:

```
<modbus enabled="True" host="0.0.0.0" port="5020">
    <device_info>
        <VendorName>Siemens</VendorName>
        <ProductCode>SIMATIC</ProductCode>
        <MajorMinorRevision>S7-200</MajorMinorRevision>
    </device_info>
    <mode>serial</mode>
    <delay>100</delay>
```

L' indirizzo dell' host viene impostato come 0.0.0.0 affinché il server ascolti su tutte le interfacce di rete disponibile, mentre la porta 5020 come visto precedentemente, viene mappata nella porta 502 dell'host al momento della creazione del *container*. In questo caso, da come si evince dall'immagine, il device simulato è il Siemens SIMATIC S7-200.

Il file **modbus.xml** è stato modificato per simulare un sistema con 3 *slave*:

- Lo *slave*_1 che mantiene una tabella per ogni tipo,
- Lo *slave*_2 che mantiene un registro di sola lettura e uno di lettura-scrittura,
- Lo *slave*_47 che mantiene tabelle per operazioni su booleani , una di lettura-scrittura e una sola lettura (figura 5.1).

Oltre ai sopraindicati slave ci sono quelli di default : lo *slave*_0 per il *broadcast* ( mantiene quindi la configurazione del Modbus seriale ) e il 255 il cui ruolo non è direttamente specificato, ma viene considerato predefinito a seconda delle politiche aziendali. Il secondo template viene usato anche dagli altri protocolli supportati da Conpot. Per ciò che concerne Modbus è qui che vengono generati i valori casuali che popoleranno i blocchi di memoria degli *slave*:

```
<slave id="47">
    <blocks>
        <block name="memoryModbusSlave47BlockA">
            <!-- COILS/DISCRETE_OUTPUTS aka. binary output, power on/power off
                 Here we map modbus addresses 1 to 127 to S7-200 PLC Addresses Q0.0 to Q15.7 -->
            <type>COILS</type>
            <starting_address>1</starting_address>
            <size>128</size>
            <content>memoryModbusSlave47BlockA</content>
        </block>
        <block name="memoryModbusSlave47BlockB">
            <!-- CONTACTS/DISCRETE_INPUTS aka. binary input.
                 Map modbus addresses 10001-10032 to S7-200 PLC inputs starting from I0.0  -->
            <type>DISCRETE_INPUTS</type>
            <starting_address>10001</starting_address>
            <size>32</size>
            <content>memoryModbusSlave47BlockB</content>
         </block>
     </blocks>
</slave>
```

Figura 5.1: Come appare la configurazione dello *slave*_47 nel file modbus.xml

```
<key name="memoryModbusSlave47BlockA">
    <value type="value">[random.randint(0,1) for b in range(0,128)]</value>
</key>
<key name="memoryModbusSlave47BlockB">
    <value type="value">[random.randint(0,1) for b in range(0,32)]</value>
</key>
```

Per sostituire i template di default con quelli personalizzati:

- docker <local_path>/modbus.xml conpot:/home/conpot/.local/lib/python3.8/site-packages/conpot/templates/default/modbus.xml
- docker cp <local_path>/template.xml conpot:/home/conpot/.local/lib/python3.8/site-packages/conpot/templates/default

Lanciamo Conpot:

- docker start conpot

### 5.2.3 Fase di testing

I test sono stati fatti usando come controparte **mbpoll**: strumento a riga di comando che permette di comunicare con dispositivi *slave* Modbus TCP o RTU. Supporta le operazioni fondamentali di lettura e scrittura sulle diverse tabelle. Per la versione TCP non sono supportate le operazioni di diagnostica (funzioni 43 e 17 ). Per questo tipo di operazioni è stato utilizzato **nmap**: un software di port scanning che permette di individuare le porte aperte su un computer o su indirizzi IP in modo da determinare quali servizi di rete siano disponibili.
L' esecuzione dei test è stata fatta su due macchine differenti nella stessa rete locale. Su entrambe le macchine è presente il sistema operativo Ubuntu versione 22.04 con la differenza che lato mbpoll viene utilizzata una macchina virtuale.
Soltanto per questi test è stato utilizzato il flag -network=host nella fase di creazione del *container*. Questo fa sì che il *container* condivida lo spazio di rete dell'host e possa essere contattato direttamente sulla porta 5020.

## 5.3 Analisi dei dati

### 5.3.1 Obiettivo

L'obiettivo di questa tesi è raccogliere dei dati per profilare le modalità di attacco degli aggressori in ambito ICS mettendo *online* un semplice honeypot. Carpire informazioni di questo genere permette di segnalare o, in presenza di un sistema più elaborato, bloccare determinati IP. Questo permette di aggiungere uno strato di sicurezza al protocollo Modbus che è nato per sistemi isolati.
Le informazioni cruciali vengono estrapolate dal file di log di Conpot per i seguenti scopi:

- 1. Studiare la distribuzione degli attacchi in base alla regione di registrazione degli IP, mettendole in relazione con i giorni della settimana e le fasce orarie in cui si verificano più frequentemente le interazioni.
- 2. Raccogliere informazioni da un database di supporto riguardo i servizi su cui si appoggiano gli utenti che interagiscono col sistema.
- 3. Verificare con il suddetto database se gli indirizzi in questione sono già stati etichettati come malevoli.
- 4. Analizzare la consapevolezza dell'utente che invia richieste: in base a queste e alla durata delle sessione è possibile fare delle supposizioni riguardo agli attacchi (e.g, se sono mirati o solo di *discovery*) e agli attaccanti (e.g., se si accorgono o meno della presenza dell' honeypot).
- 5. Comprendere dal file quali sono le operazioni maggiormente richieste agli slave Modbus per fare delle assunzioni rispetto alle intenzioni degli attaccanti: se il comportamento è quello che ci si aspetta e, quindi, vengono inviati operazioni di lettura/scrittura oppure è più comune che Conpot generi errori o eccezioni.
- 6. Nel caso gli IP non siano stati etichettati come malevoli, confrontare i comportamenti degli utenti che sfruttano IP che sembrano innocui con quelli profilati come malevoli o comunque, verificare che ci siano delle attività anomale.
- 7. Analizzare il numero delle sessioni avviate dallo stesso IP in giornate diverse o consecutivamente. Questo potrebbe essere utile per capire se viene usato un tool di *scanning* della rete o se l' utente prova semplicemente più volte a collegarsi al sistema.
- 8. Con le informazioni raccolte cercare di riconoscere dei pattern per sviluppare una tecnica di *finger-printing* specifica per Modbus.
- 9. Monitorare gli eventuali comportamenti anomali di Conpot e cercare di studiarne la causa.

### 5.3.2 Deployment

#### Ambiente

Per esporre Conpot e raccogliere dati reali è stato utilizzato un servizio **Iaas** (*Infrastructure as a Service*) che permette l' accesso *on-demand* a risorse informatiche. In questo specifico caso tali risorse sono : un Ubuntu server versione 22.04.03 sulla quale gira Conpot su Docker (quindi con la porta 502 esposta) e un Windows Server 2022 di appoggio nello stesso network.

#### Dati Raccolti

Quelli che vengono presentati in seguito sono delle statistiche sui dati raccolti usando Conpot con i template brevemente descritti sopra su un sistema ubicato in Italia.
I dati considerati fanno riferimento al periodo dal 21-02-2024 al 13-03-2024 incluso.
Il primo studio interessante fatto è quello della distribuzione per aree geografiche (figura 5.2).
Si noti che la geo-localizzazione basata solamente sugli indirizzi IP non è affidabile, gli attaccanti aggirano questo metodo usando *proxy*, VPN o dislocando i sistemi in diversi paesi [49].

| Paese | % IP per aree geografiche |
|---|---|
| Stati Uniti | 77,3% |
| Cina | 11,8% |
| Regno Unito | 4,2% |
| Germania | 3,4% |
| Belgio | 1,7% |
| Olanda | 0,8% |
| Costa Rica | 0,8% |

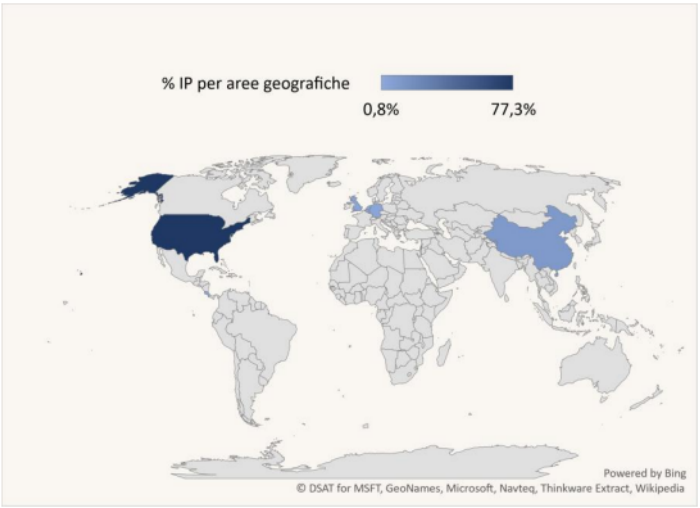


Figura 5.2: Distribuzione IP per aree geografiche

Nel grafico 5.3 viene visualizzato l' andamento delle interazioni per giorno, intese come intere sessioni avviate e non come singoli comandi.
Alcuni indirizzi IP sono duplicati ma solamente in giornate diverse.

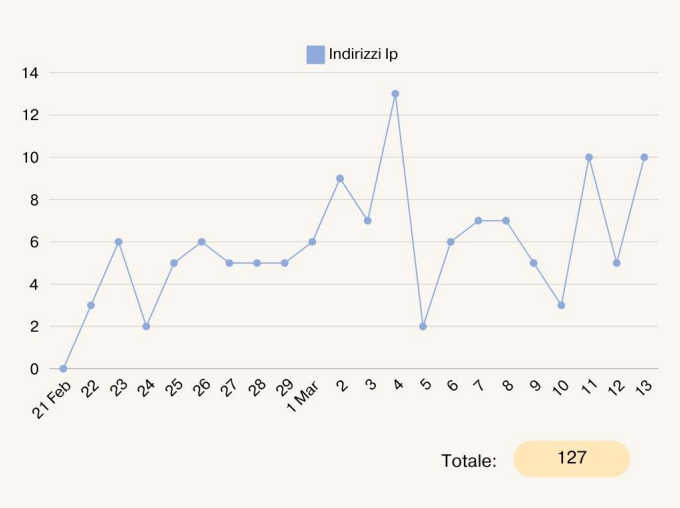


Figura 5.3: Interazioni per giorno

In seguito (grafico 5.4) si è voluto mettere in relazione i giorni della settimana al paese d'origine dell'IP. Il fine è verificare se ci sono giorni in cui le interazioni sono più comuni e se varia qualcosa da paese a paese.

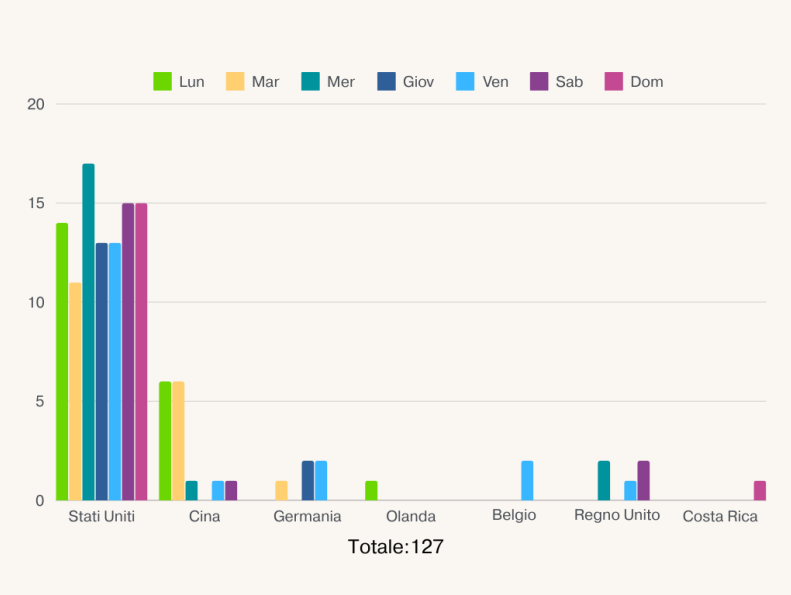


Figura 5.4: Giorni della settimana in cui si sono verificate delle interazioni

Per lo stesso principio descritto sopra, il grafico 5.5 raccoglie fasce orarie in cui sono avvenute le interazioni e anche qui sono state messe in corrispondenza con i paesi listati prima.

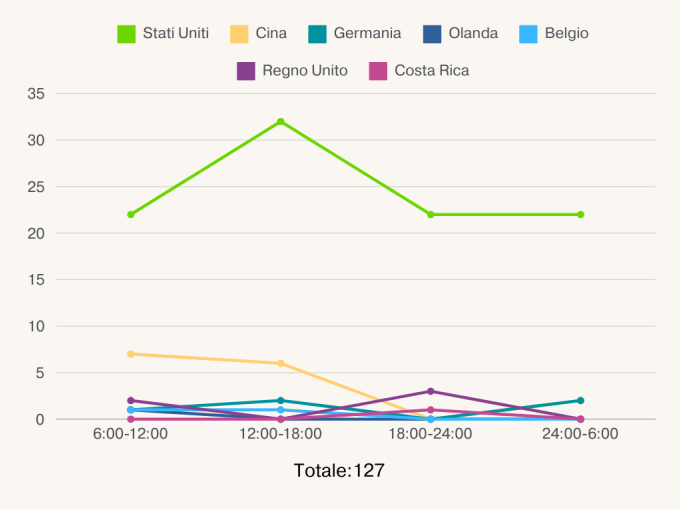


Figura 5.5: Frequenza fasce orarie in relazione al paese d'origine dell' IP

Il grafico 5.6 raccoglie i servizi rilevati da un database di IP (***AbuseIPDB***).

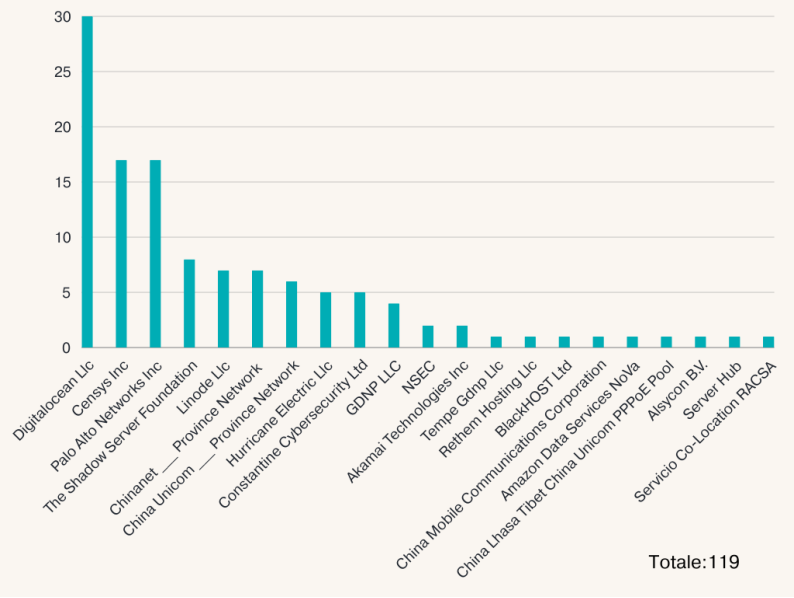


Figura 5.6: Servizi

Un altro aspetto che si ritiene opportuno analizzare è quello legato alle percentuali di IP malevoli, non malevoli e presunti tali che hanno interagito col sistema. Sfruttando *AbuseIPDB* è stato possibile venire a conoscenza della percentuale di affidabilità dei

singoli indirizzi.
Ne è risultato che è possibile suddividere i suddetti in 4 diverse categorie:

- gli IP malevoli,
- gli IP con un' alta probabilità che siano malevoli,
- gli IP con bassa probabilità che siano malevoli,
- gli IP non malevoli, secondo il database.

Nel grafico ad anello 5.7 viene raffigurata questa configurazione, considerando gli indirizzi IP senza duplicati.

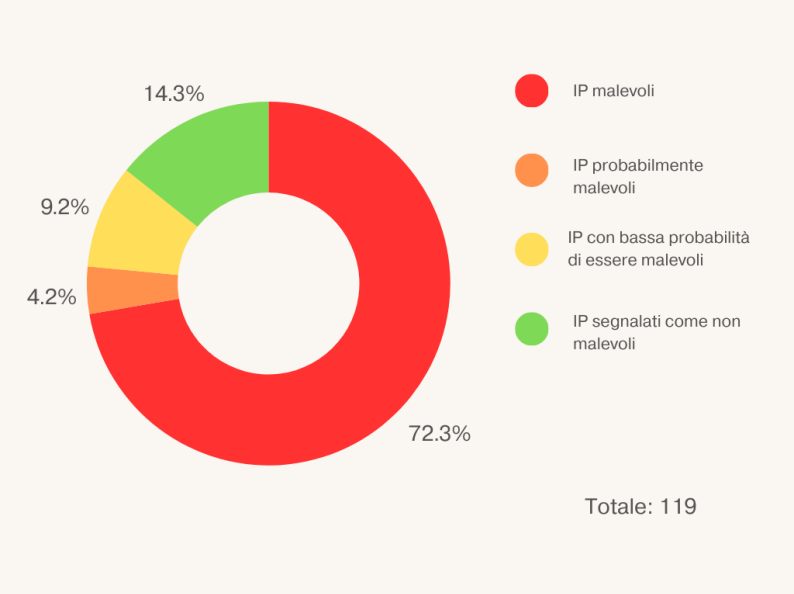


Figura 5.7: Categorie IP e percentuali

La durata delle sessioni potrebbe essere un altro elemento da considerare.
Nel grafico 5.8 viene sintetizzata la situazione durante il periodo di attività dell'honeypot. Si noti che che le barre in rosso indicano che la relativa interazione si è conclusa con un malfunzionamento di Conpot. In questi casi è stato necessario riavviare il *container* poiché gli attaccanti erano riusciti a raggirare la normale durata del timer di sessione ( 32 secondi ca ), non permettendone la scadenza e nemmeno la connessione da parte di altri utenti. Le barre nere indicano quegli eventi per cui non è stata registrata la scadenza del time-out di sessione. Le sessioni sono 128 perché 2 interazioni consecutive fatte dallo stesso IP sono state considerate separate (quindi con identificativo differente) a causa della scadenza del time-out.
Per maggiore leggibilità si omette la numerazione sulle ascisse.

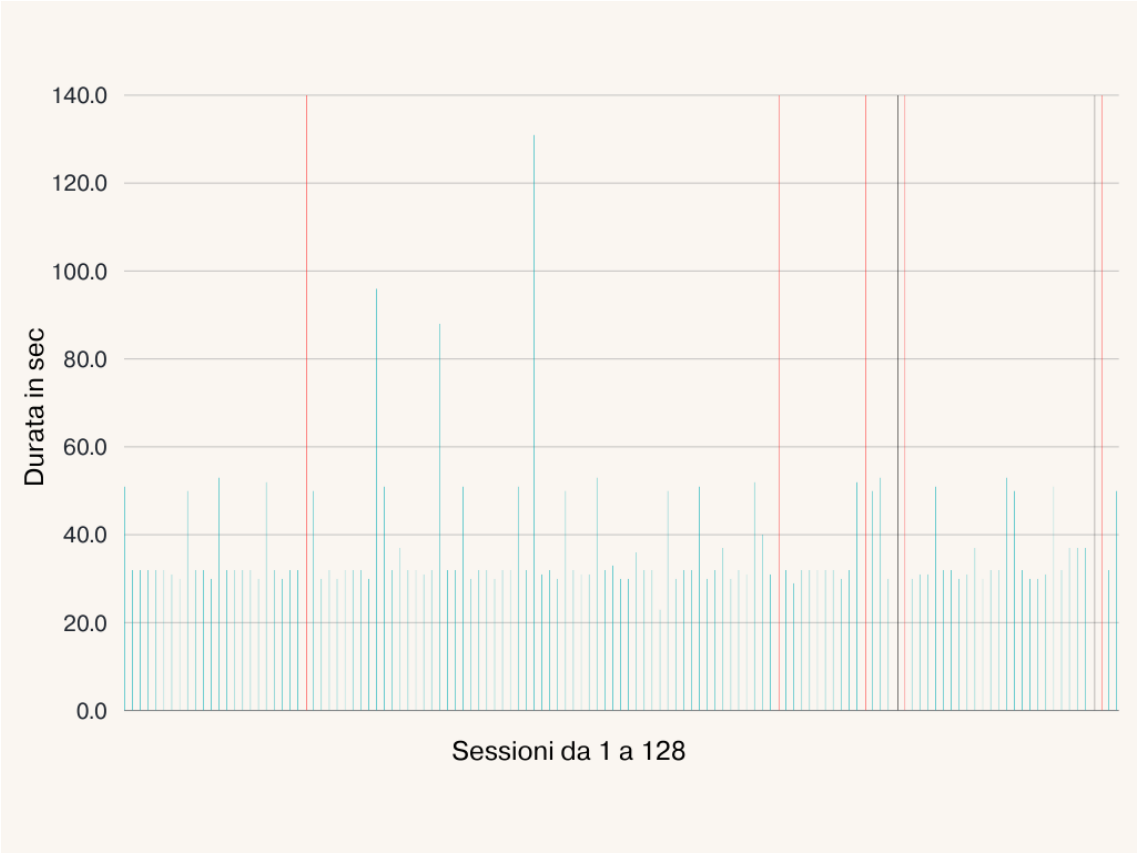


Figura 5.8: Durata sessioni in secondi

Nella tabella 5.9 sono riportate informazioni del tipo: numero successione relativo al grafico 5.8, data, indirizzo IP e durata della sessione riguardo le interazioni che superano il tempo standard del timer. I malfunzionamenti in corrispondenza della barre rosse vengono indicate con la scritta inf, mentre gli eventi di cui non si conosce il *timestamp* di fine sessione con ?.

| | Data | Indirizzo IP | Durata in sec | | Data | Indirizzo IP | Durata in sec |
|---|---|---|---|---|---|---|---|
| 1 | 02-22 14:22:24 | 167.94.146.XX | 51 | 94 | 03-08 21:20:16 | 167.94.145.XX | 52 |
| 9 | 02-23 18:55:08 | 162.142.125.XXX | 50 | 95 | 03-08 23:10:00 | 87.236.176.XXX | inf |
| 13 | 02-25 06:02:10 | 167.94.145.XX | 53 | 96 | 03-09 17:51:59 | 167.94.138.XXX | 50 |
| 19 | 02-26 08:36:25 | 167.94.146.XX | 52 | 97 | 03-09 18:34:22 | 162.142.125.XX | 53 |
| 24 | 02-27 01:12:47 | 172.105.246.XXX | inf | 99 | 03-09 22:30:03 | 87.236.176.XX | ? |
| 25 | 02-27 10:58:32 | 167.94.138.XX | 50 | 100 | 03-09 22:30:23 | 87.236.176.XXX | inf |
| 33 | 02-29 01:35:09 | 172.105.246.XXX | 96 | 104 | 03-11 13:25:07 | 167.94.145.XX | 51 |
| 34 | 02-29 03:27:10 | 167.94.146.XX | 51 | 109 | 03-11 16:55:50 | 219.140.42.XXX | 37 |
| 36 | 02-29 08:15:42 | 104.152.52.XXX | 37 | 113 | 03-12 03:06:39 | 167.94.146.XX | 53 |
| 41 | 03-01 15:45:10 | 64.225.104.XXX | 88 | 114 | 03-12 10:23:51 | 167.94.138.XX | 50 |
| 44 | 03-02 00:15:27 | 167.94.145.XX | 51 | 119 | 03-13 05:13:34 | 167.94.145.XX | 51 |
| 51 | 03-02 17:06:39 | 167.94.146.XX | 51 | 121 | 03-13 07:05:31 | 172.104.210.XXX | 37 |
| 53 | 03-03 05:57:50 | 36.156.22.X | 131 | 122 | 03-13 07:20:46 | 198.74.56.XX | 37 |
| 57 | 03-03 13:05:25 | 167.94.138.XXX | 50 | 123 | 03-13 07:38:28 | 45.79.163.XX | 37 |
| 61 | 03-04 05:58:41 | 167.94.138.XXX | 53 | 124 | 03-13 09:44:53 | 87.236.176.XXX | ? |
| 66 | 03-04 06:30:49 | 182.138.158.X | 36 | 125 | 03-13 09:45:13 | 87.236.176.XXX | inf |
| 70 | 03-04 11:27:38 | 162.142.125.XXX | 50 | 127 | 03-13 17:19:39 | 167.94.138.XXX | 50 |
| 74 | 03-05 15:24:20 | 167.94.146.XX | 51 | | | | |
| 77 | 03-06 07:15:53 | 192.155.88.XXX | 37 | | | | |
| 81 | 03-06 20:02:19 | 162.142.125.XXX | 52 | | | | |
| 82 | 03-07 01:35:59 | 167.94.138.XXX | 40 | | | | |
| 84 | 03-07 10:34:35 | 138.68.88.XX | inf | | | | |

Figura 5.9: Dettagli delle interazioni più lunghe

In aggiunta all'analisi delle sessioni è utile esaminare gli eventi registrati da Conpot, queste informazioni potrebbero essere servire a verificare la presenza di pattern o punti comuni nelle interazioni.
Le operazioni effettuate dagli utenti dopo la connessione con Conpot sono sommariamente sempre le stesse, quello che varia è la loro frequenza.
I tipi di eventi registrati da Conpot sono:

- 1. Errori generati dalla richiesta dell'operazione 43, 17 o 3 allo slave 0. Queste operazioni non sono valide poiché nel protocollo Modbus implementato su Conpot non è prevista la risposta alle richieste di lettura quando si opera in *broadcast*.
- 2. Errore 104: *connection reset by peer*
- 3. Mbap non valido quindi pacchetti malformati
- 4. Errori legati alla struttura del buffer: *struct.error: unpack requires a buffer of 1 bytes*

- 5. Eventi di ricezione dati non validi
- 6. Eccezioni di *time-out* da parte del *server protocol* Modbus che indicano che l'utente si è disconnesso prima della scadenza del timer
- 7. Altre operazioni che non generano errori o eccezioni :
  - Operazione 17 su *slave* diversi da 0
  - Operazione 43 su *slave* diversi da 0
  - Operazione *none* sullo *slave* 255

Nella figura 5.10 viene mostrata la frequenza delle operazioni sopracitate.
I numeri nel grafico fanno riferimento alla tipologia di evento esplicitata sopra.

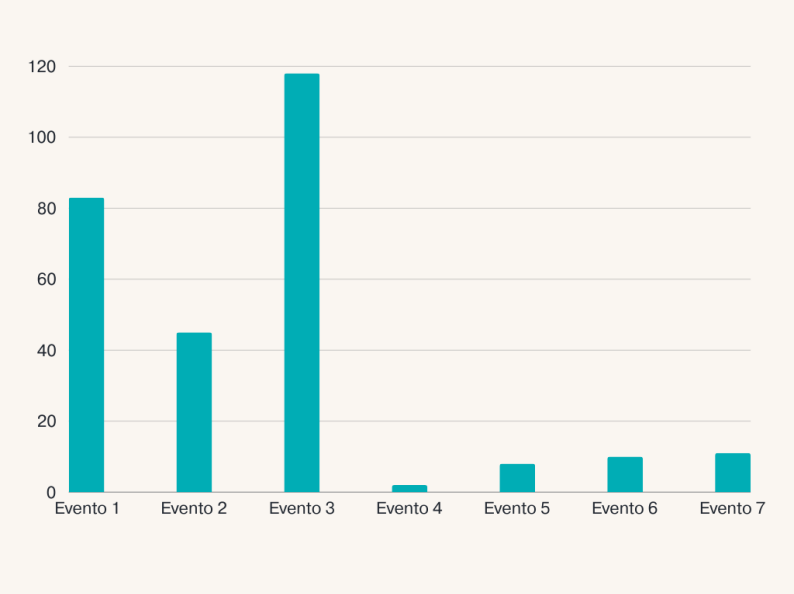


Figura 5.10: Frequenza con cui si sono verificati gli eventi

Considerando l'evento 3, talvolta anche all'interno della stessa interazione vengono mandati pacchetti malformati: la risposta Modbus generata non corrisponde alla lunghezza prevista.
Nei dati raccolti ci sono eventi di questo genere che si verificano consecutivamente. La serie più lunga è di 11 *"modbus_tk.modbus_tcp.ModbusInvalidMbapError"* a distanza di pochi secondi l'una dall'altra.
Nel grafico 5.11 viene indicato il totale degli eventi con frequenza più significativa (1, 2, 3) per IP malevoli.
Si prendono in considerazione gli indirizzi correlati ad almeno uno dei tre eventi con frequenza totale $> 1$.

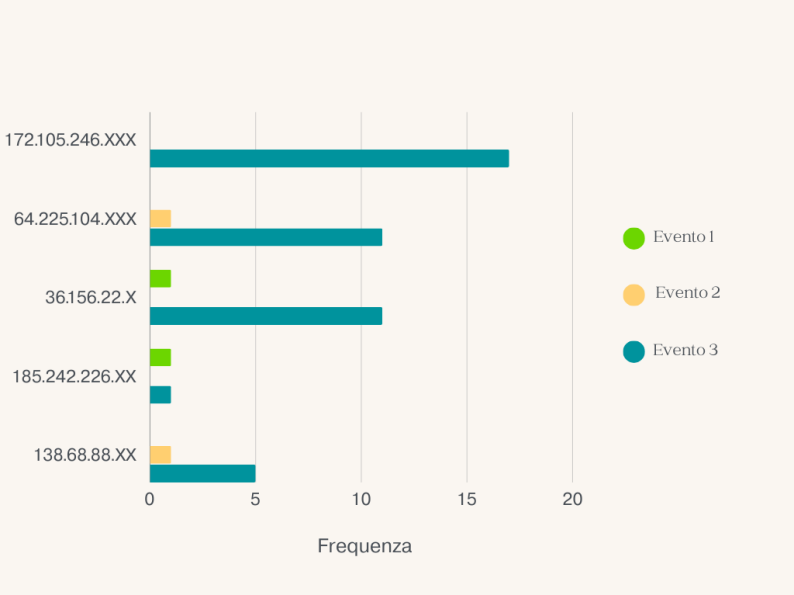


Figura 5.11: Frequenza degli eventi più significativi nelle sessioni di IP malevoli

Non si approfondisce ulteriormente la frequenza degli eventi generati da IP targhettati come sicuramente non malevoli. Questa scelta deriva dal fatto che questi sono tutti IP Censys e simili. Vengono usati dei tool automatici per questo scopo ed infatti il loro comportamento è simile : gli eventi generati sono perlopiù di tipo 1, 2 e 3.
Per gli obiettivi indicati sopra è invece interessante analizzare il comportamento degli IP di dubbia provenienza con basse percentuali di segnalazioni all'epoca dei fatti (tabella 5.12).

| Data | Orario | IP | FC | Slave | Evento | Provenienza |
|---|---|---|---|---|---|---|
| 23-02 | 05:13:12 | 104.206.128.XX | 43 | 0 | ERROR:conpot.protocols.modbus.slave_db:Function 43 can not be broadcasted | Stati Uniti |
| 29-02 | 08:15:42 | 104.152.52.XXX | | | ERROR:conpot.protocols.modbus.modbus_server:Exception occurred in ModbusServer.handle() at sock.recv(): timed out | Stati Uniti |
| 03-03 | 07:34:54 | 44.220.185.XX | 43 | 0 | ERROR:conpot.protocols.modbus.slave_db:Function 43 can not be broadcasted | Stati Uniti |
| 04-03 | 06:30:31 | 119.48.135.XXX | 43 | 0 | ERROR:conpot.protocols.modbus.slave_db:Function 43 can not be broadcasted | Cina |
| | 06:30:32 | 119.164.104.XXX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 60929 while receiving 237 bytes. | Cina |
| | 06:30:37 | 125.82.242.XXX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 60929 while receiving 237 bytes. | Cina |
| | 06:30:43 | 218.66.32.XXX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 51 bytes. | Cina |
| | 06:30:49 | 182.138.158.X | | | ERROR:conpot.protocols.modbus.modbus_server:Exception occurred in ModbusServer.handle() at sock.recv(): timed out | Cina |
| | 06:30:55 | 27.98.228.XXX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 8289 while receiving 10 bytes. | Cina |
| 10-03 | 21:28:17 | 190.10.8.XXX | | | New Modbus connection from 190.10.8.XXX | Costa Rica |
| | | | | | New Modbus connection from 190.10.8.XXX | |
| | | | | | New Modbus connection from 190.10.8.XXX | |
| | | | 43 | 0 | ERROR:conpot.protocols.modbus.slave_db:Function 43 can not be broadcasted | |
| 11-03 | 16:55:29 | 223.199.181.XX | 43 | 0 | ERROR:conpot.protocols.modbus.slave_db:Function 43 can not be broadcasted | Cina |
| | 16:55:31 | 180.95.231.XX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 60929 while receiving 237 bytes. | Cina |
| | 16:55:37 | 101.68.127.XX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 60929 while receiving 237 bytes. | Cina |
| | 16:55:44 | 223.199.181.XXX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 51 bytes. | Cina |
| | 16:55:50 | 219.140.42.XXX | | | ERROR:conpot.protocols.modbus.modbus_server:Exception occurred in ModbusServer.handle() at sock.recv(): timed out | Cina |
| | 16:55:55 | 61.158.26.XXX | | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 8289 while receiving 10 bytes. | Cina |
| | | | | | | |
| | FC= FUNCTION CODE | | | | | |

Figura 5.12: Sequenza di eventi generati da IP non targhettati come malevoli

Osservando la tabella si sono quindi verificate delle connessioni consecutive che sem-

brano essere attività anomale: alcune provengono da IP diversi ed altre da IP uguali. La stessa situazione è stata riscontrata il 29-02: sono state registrate 5 nuove connessioni consecutive dallo stesso indirizzo IP malevolo: 172.105.246.XXX in meno di un secondo. Lo stesso poco dopo ne ha tentate altre 2.
Dal registro è stato anche rilevato che due IP malevoli hanno tentato 2 interazioni in giorni diversi:

- 184.105.247.XXX
- 172.105.246.XXX

In diverse giornate, come è stato anticipato precedentemente, si sono verificati dei malfunzionamenti di Conpot. Quelli del 27-02 (tabella 5.13 ) e del 7-03 (tabella 5.14) con IP registrati in Germania mentre, gli altri 3 con IP provenienti dal Regno Unito (tabella 5.15).

| Data | Ora | Indirizzo IP | Evento |
|---|---|---|---|
| 27-02 | 01:12:47 | 172.105.246.XXX | ERROR:conpot.protocols.modbus.modbus_server:Exception occurred in ModbusServer.handle() at sock.recv(): timed out |
| | 01:12:52 | | Modbus client provided data 1c326be4-eed8-4d01-a4fa-72d8cc4ca4b9 but invalid. |
| | 01:12:52 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 12 bytes. |
| | 01:12:57 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 20302 while receiving 16 bytes. |
| | 01:13:02 | | New Modbus connection |
| | | | Riavvio container |

Figura 5.13: Presunto attacco del 27-02

| | Ora | Indirizzo IP | Evento |
|---|---|---|---|
| 07-03 | 10:34:35 | 138.68.88.XX | ERROR:conpot.protocols.modbus.modbus_server:Exception occurred in ModbusServer.handle() at sock.recv(): [Errno 104] Connection reset by peer |
| | 10:34:40 | | ERROR:conpot.protocols.modbus.modbus_server:Exception occurred in ModbusServer.handle() at sock.recv(): timed out |
| | 10:34:45 | | Modbus client provided data a83ef9ce-d8a0-4d59-9326-1b853693d465 but invalid. |
| | 10:34:45 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 12 bytes. |
| | 10:34:50 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 20302 while receiving 16 bytes. |
| | 10:34:55 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 20302 while receiving 16 bytes. |
| | 10:35:00 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 29438 while receiving 38 bytes. |
| | 10:35:05 | | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 256 while receiving 26 bytes. |
| | 10:35:10 | | Modbus client provided data a83ef9ce-d8a0-4d59-9326-1b853693d465 but invalid. |
| | 10:35:10 | | New modbus connection |
| | | | Riavvio container |

Figura 5.14: Presunto attacco del 7-03

| | Ora | Indirizzo IP | Evento |
|---|---|---|---|
| 08-03 | 23:10:00 | 87.236.176.XXX | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 192 bytes. |
| | 23:10:20 | | New modbus connection |
| | | | Riavvio container |
| | Ora | Indirizzo IP | Evento |
| 09-03 | 22:30:03 | 87.236.176.XX | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 192 bytes. |
| | 22:30:23 | 87.236.176.XXX | New modbus connection |
| | | | Riavvio container |
| | Ora | Indirizzo IP | Evento |
| 13-03 | 09:44:53 | 87.236.176.XXX | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 12064 while receiving 192 bytes. |
| | 09:45:13 | 87.236.176.XXX | New Modbus connection from 87.236.176.195 |
| | | | Riavvio container |

Figura 5.15: Presunto attacco da parte di IP del Regno Unito

### Confronti con altri dati

Durante questo periodo di raccolta dati ci si è chiesto se cambiasse qualcosa modificando l' ambiente di Conpot, per cui sono state fatte due prove ulteriori ignorando gli IP sicuramente non malevoli. Nelle prossime righe l'honeypot già trattato verrà denominato honeypot_1 e di conseguenza il corrente honeypot_2.

La prima prova è stata fatta utilizzando lo stesso *template* descritto sopra installando Conpot in un ambiente casalingo su una macchina Ubuntu versione 22.04.
I dati riportati di seguito fanno riferimento al periodo 11-03-2024/18-03-2024.
Nella figura 5.16 vediamo come gli IP registrati negli Stati Uniti siano sempre in numero maggiore.

| Paese | % IP per aree geografiche |
|---|---|
| Stati Uniti | 73,1% |
| Cina | 11,5% |
| Germania | 3,8% |
| Olanda | 3.8% |
| Canada | 3.8% |
| Corea | 3,8% |

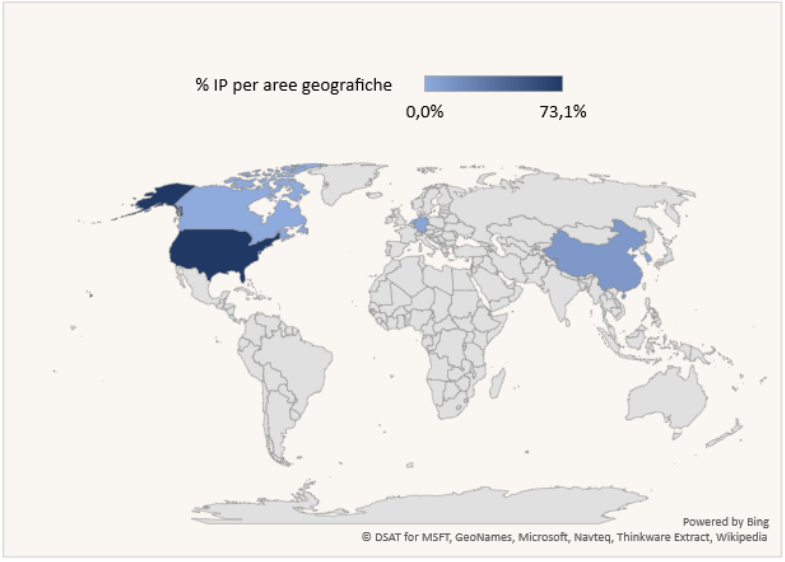


Figura 5.16: Distribuzione IP per aree geografiche dell' honeypot_2

Il servizio più usato anche in questo caso risulta essere la tecnologia della Digital Ocean (figura 5.17).

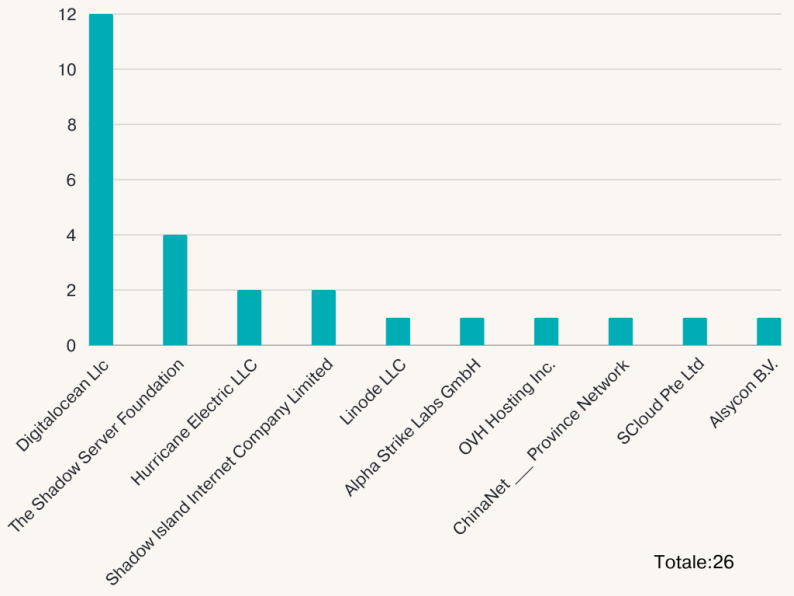


Figura 5.17: Servizi utilizzati nell'honeypot_2

Anche nell'honeypot_2 la maggioranza è rappresentata dagli IP malevoli (figura 5.18).

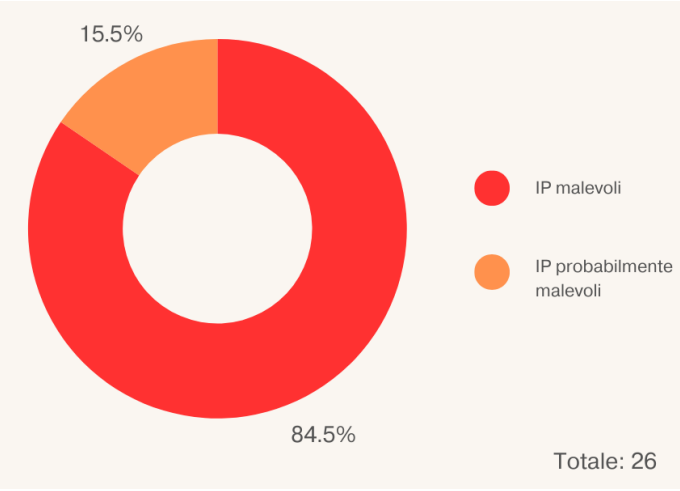


Figura 5.18: Categorie IP e percentuali presenti nelle sessioni dell'honeypot_2

Tre IP erano già stati visti nell'honeypot_1:

- 198.199.119.XXX

- 162.243.138.XX
- 185.242.226.XX

I loro comportamenti sono molto simili all'analisi fatta antecedentemente. Si è potuto apprendere che la situazione non è molto differente rispetto al caso dell' honeypot_1. Cambia la distribuzioni degli IP nei paesi e di conseguenza anche alcuni dei servizi usati, ma le operazioni effettuate sono perlopiù situazioni che generano eventi ed eccezioni simili all'esperimento fatto prima. L'unica differenza riguardo le operazioni è il tentativo di contattare lo slave_3 che è inesistente. In questi casi, come ci si aspetta, viene generato un errore.

Nella seconda prova Conpot gira su una macchina virtuale Ubuntu 22.04 con ubicazione in Svizzera. I dati registrati si riferiscono al periodo che va dal 16-03-2024 al 23-03-2024. In questo caso è stato usato un template che simula un S7-1500 con 6 slave.
Gli IP:

- 162.243.132.XX
- 172.104.11.XX
- 172.104.210.XXX
- 45.79.163.XX
- 74.82.47.X
- 142.4.218.XXX

erano già presente nel file di log degli honeypot precendenti e anche in questo caso il loro comportamento è il medesimo.
Mentre gli IP:

- 172.105.246.XXX
- 65.49.20.XX

non fanno esattamente la stessa cosa. Il primo IP è presente nella tabella 5.9, una delle due volte in cui si è collegato a Modbus ha causato un malfunzionamento. Ma se analizziamo la sequenza di eventi ci rendiamo conto che nel caso dell'honeypot_3 il numero di comandi inviati è molto più alto. L' altro IP invece, nel caso dell'honeypot_2, inviò un'operazione valida sullo slave_3 mentre in questo caso ha generato un evento di tipo 1.
Nella figura 5.19, come è ormai noto, vediamo che gli Stati Uniti sono nuovamente in testa per il numero di interazioni fatte con gli honeypot.

| Paese | % IP per aree geografiche |
|---|---|
| Stati Uniti | 68,6% |
| Cina | 11,4% |
| Olanda | 5,7% |
| Germania | 5,7% |
| Regno Unito | 2,9% |
| Giappone | 2,9% |
| Canada | 2,9% |

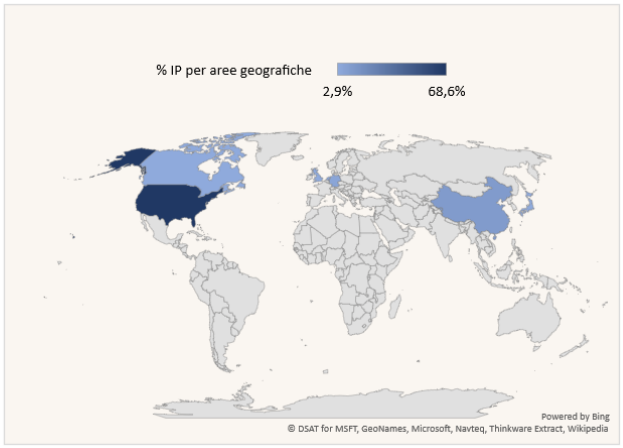


Figura 5.19: Distribuzione IP per aree geografiche dell' honeypot_3

Come ci si aspetta Digital Ocean è il servizio più comune (figura 5.20).

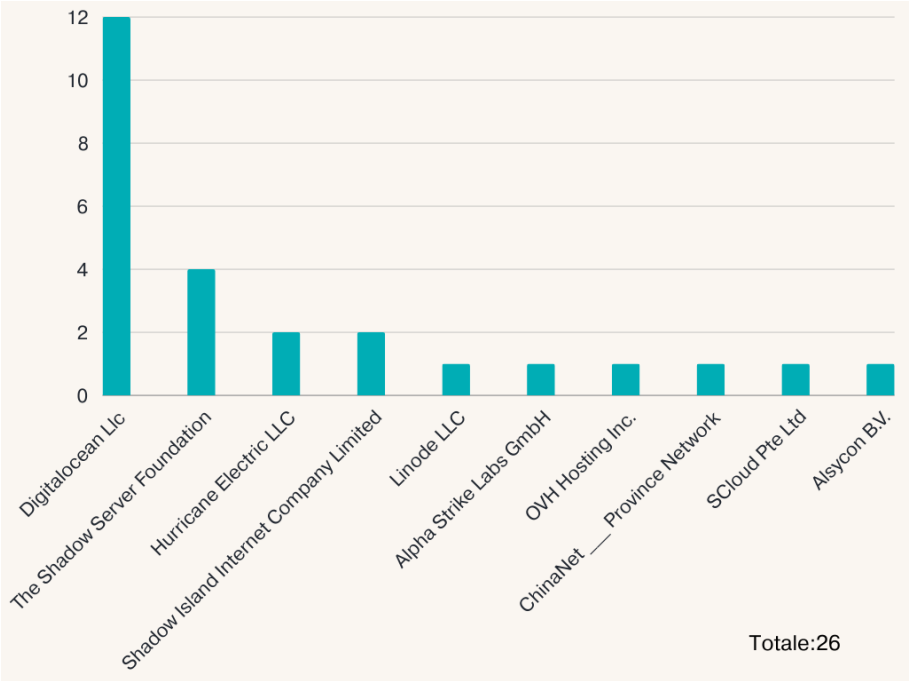


Figura 5.20: Servizi utilizzati nell'honeypot_3

Ancora una volta gli IP malevoli rappresentano la maggioranza rispetto a quelli con più bassa probabilità di esserlo (figura 5.21).

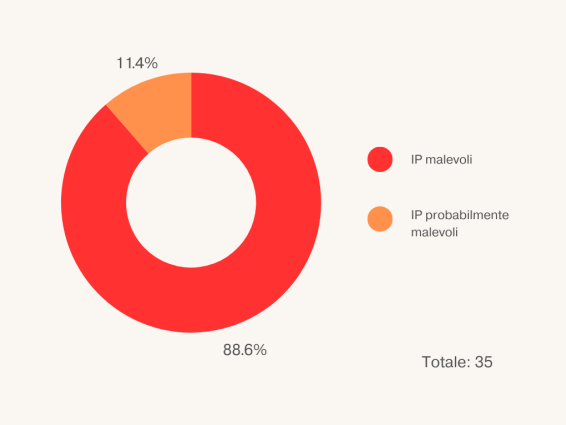


Figura 5.21: Categorie IP e percentuali presenti nelle sessioni dell'honeypot_3

Ci sono degli aspetti singolari da analizzare.
Il 16-03 è stata rilevata una sessione di 6 minuti e 56 secondi, una durata insolitamente alta se facciamo riferimento alla tabella 5.9. L' indirizzo in questione risulta essere registrato nel Regno Unito.
Per questioni di spazio, gli eventi noti saranno indicati con la numerazione precedentemente presentata. Si noti che in arancione vengono sottolineati eventi che non si erano ancora verificati negli altri honeypot (tabella 5.22).

| Ora | Indirizzo IP | FC | Slave | Conpot response / ERROR / Exception |
|---|---|---|---|---|
| 10:23:36 | 165.154.129.XXX | | | Evento_6 |
| 10:23:43 | | | | Evento_3 |
| 10:23:51 | | 43 | 0 | Evento_1 |
| 10:23:53 | | | | Evento_5 |
| Dalle 10.23.55 alle 10:24:59 | | | | Evento_3 (x 10 volte) |
| 10:25:01 | | | | Evento_5 |
| 10:25:03 | | | | Evento_3 |
| 10:25:10 | | | | Evento_5 |
| Dalle 10:25:12 alle 10:26:18 | | | | Evento_3 (x 11 volte) |
| 10:26:25 | | none | 2 | ERROR:conpot.protocols.modbus.slave_db:Slave 2 doesn't exist |
| Dalle 10:26:28 alle 10:27:58 | | | | Evento_3 ( x 15 volte) |
| 10:28:05 | | | 50 | raise MissingKeyError("Slave {0} doesn't exist".format(slave_id)) |
| | | | | During handling of the above exception, another exception occurred: |
| | | | | TypeError: unsupported operand type(s) for +: 'NoneType' and 'int' |
| Dalle 10:28:07 alle 10:29:16 | | | | Evento_3 (x 15 volte) |
| 10:29:23 | | | | Exception code = 1 (x 8 volte) |
| 10:29:23 | | 0 | 0 | |
| | | | | Evento_3 (x 16 volte) |
| | FC=Function Code | | | |

Figura 5.22: Sessione del 16-03 nel dettaglio

Mentre durante le altre volte in cui si è verificato un interruzione del servizio sono state sollevate delle eccezioni (5.13, 5.14, 5.15 ), questa volta oltre alla connessione

del dispositivo non viene registrato nient'altro nei minuti precedenti. L'evento antecedente risale a più di 5 ore prima e proviene da un IP registrato negli Stati Uniti (tabella 5.23).

| | Ora | Indirizzo IP | Evento |
|---|---|---|---|
| 17-03 | 02:03:06 | 107.170.226.XX | modbus_tk.modbus_tcp.ModbusInvalidMbapError: Response length is 17476 while receiving 19 bytes. |
| | 07:45:27 | 106.75.174.XXX | New Modbus connection from 106.75.174.XXX |
| | | | Riavvio container |

Figura 5.23: Malfunzionamento dell'honeypot_3 in seguito all' interazione con un IP cinese

Il 23-03 viene causato un malfunzionamento da un IP che non ha probabilità 100% di essere malevolo. Come visto in casi analizzati prima anche in questa circostanza vengono forniti dati non validi e viene generato un errore. Quest'ultimo viene causato da un *invalid Mbap* dove la lunghezza della risposta generata è di lunghezza 0 ma la richiesta risultava essere di 1 byte.

Facendo invece riferimento ad altri lavori, viene confermato ancora una volta il fatto che gli Stati Uniti producono più traffico rispetto agli altri paesi e che coloro che hanno utilizzato codici funzioni validi sono molto meno rispetto alle operazioni tipiche di Modbus [50].

# Capitolo 6

# Considerazioni finali

Per rispondere al punto (1) della sezione "Obiettivi", considerando i grafici 5.2, 5.4 5.5 e osservando soltanto i paesi più influenti ( di cui abbiamo più dati ), possiamo concludere che:

- Il periodo di attività maggiore degli IP localizzati negli Stati Uniti si colloca il mercoledì nella fascia oraria 12:00-18:00.
- Per quanto riguarda la Cina, i giorni della settimana in cui avvengono più interazioni sono il lunedì e il martedì dalle 6:00 alle 12:00.
- Per il Regno Unito i giorni più comuni sono il mercoledì e il sabato e l'orario si aggira intorno alle 18:00-24:00.
- La Germania è più attiva il giovedì e il venerdì nelle fasce orarie 12:00-18:00/24:00-6:00.

Globalmente, il giorno più bersagliato è stato lunedì 3 Marzo 5.3.
Molti IP provengono dagli Stati Uniti, come detto in precedenza ciò non implica che le macchine dalla quale derivano gli attacchi siano necessariamente ubicate negli Stati Uniti ma potrebbe essere usato come spunto di riflessione. Il primo motivo, già discusso, è legato al voler nascondere la reale posizione dell'attaccante ma questo ha come conseguenza anche l'alterazione di statistiche. Molti attacchi attribuiti agli Stati Uniti potrebbero in realtà provenire da altri paesi che compaiono meno frequentemente in un report. In questo caso c'è una disparità tale da far pensare che molti IP siano localizzati in posti diversi rispetto alla loro reale posizione. Se questo non fosse vero bisognerebbe approfondire perché c'è un'alta concentrazione di attacchi provenienti dagli Stati Uniti: se è una questione legata a mancanza di restrizioni che sono presenti invece in altri paesi o l'ampia presenza di server e risorse che facilitano gli attacchi.
Le risposte ai punti (2) e (3) derivano direttamente dalle statistiche visibili nei grafici 5.6 e 5.7.
I casi in cui sono state avviate sessioni più lunghe fanno pensare ad attacchi manuali mirati. Se analizziamo attentamente l' attacco della tabella 5.22 ci rendiamo

conto che è probabile che l' attaccante conoscesse il protocollo Modbus e le eccezioni che sono fonte di vulnerabilità in esso. Infatti è l'unico caso in cui viene sollevata l'*Exception Code:1*, ovvero l' *Illegal Function Exception*, protagonista della vulnerabilità presentata a pagina 24. Anche i pochi casi in cui vengono richieste operazioni valide in Modbus danno l'impressione di essere state fatte ad hoc.
La situazione sembra diversa se analizziamo le sessioni di breve durata, hanno tutto l' aspetto di essere state fatte per scopi di *discovery* vista la somiglianza delle interazioni.
Per quanto riguarda la consapevolezza della presenza dell'honeypot, anche se non immediatamente, l'honeypot_1 è stato rilevato da Shodan per cui è possibile che, anche se non ci sono state differenze sostanziali nelle interazioni, alcuni comandi fossero mirati a mettere l' honeypot_1 fuori uso. C'è da aggiungere però, che un' attaccante consapevole della presenza di un honeypot dovrebbe cercare di lasciare meno tracce possibili e quindi evitare totalmente l' interazione o raggirarla (punto (4)). A questo proposito si vuole sottolineare che con un template più complesso che simula un device diverso da quello standard Conpot ( caso honeypot_3 ), Shodan all'epoca dei fatti non aveva rilevato la presenza del suddetto. La risposta al punto (5) si basa sui dati raccolti nella tabella 5.10. Il comportamento è molto diverso da quello che ci si aspettava poiché erano state previste operazioni di scrittura o che sfruttassero i punti deboli noti del protocollo Modbus. Mettendo in relazione gli IP della figura 5.11 alla durata delle sessioni più lunghe (figura 5.9) è venuto fuori:

- IP: 172.105.246.XXX, numerazione tabella: 24, 43
- IP: 64.225.104.XXX, numerazione tabella: 41
- IP: 36.156.22.X, numerazione tabella: 53

Gli eventi più comuni di alcune delle sessioni più lunghe sono quelli del tipo evento_3.
Dalle osservazioni fatte riguardo al numero sproporzionato di eventi che generano errori rispetto alle operazioni Modbus attese, viene dedotto che il motivo principale degli aggressori è quello di eseguire il *finger-printing* per raccogliere informazioni piuttosto che sfruttare direttamente le vulnerabilità dei protocolli dei sistemi di controllo industriali [50].
Possiamo fare delle supposizioni riguardo alle informazioni a cui un aggressore potrebbe essere interessato:

- La ricerca di una possibile cattiva configurazione del device, ad esempio legati alla configurazione dello slave_0 : se fosse configurato in maniera scorretta potrebbe provocare *flooding* di dati. Infatti, nel caso in cui molti slave rispondessero alla richiesta ci potrebbe essere un impiego inutile di risorse.
- L'analisi del buffer. L'invio di pacchetti malformati potrebbe essere legata alla volontà di comprendere la capacità del buffer: se si riuscisse a saturarlo si

potrebbe sfruttare la vulnerabilità del buffer *overflow* presentato nel capitolo dedicato alle vulnerabilità Modbus.

Per rispondere al punto (6) si faccia riferimento alla figura 5.12. I casi certamente con intenti non benigni sono quelli con svariate connessioni a pochi secondi di distanza. Nel caso della Cina ci sono due casi molto simili tra loro in cui, anziché, usare stesso IP vengono fatti nuovi accessi da IP diversi tra loro tutti con bassa probabilità di essere malevoli. Usando diversi indirizzi il numero di segnalazioni si distribuisce riuscendo così a mantenere delle percentuali di pericolosità basse. Un altro caso che non lascia spazio a immaginazione è quello del 23-03, dove un IP non sicuramente malevolo ha generato uno stop di servizio dell'honeypot_3. Per quanto riguarda le attività sospette degli altri indirizzi IP, seppur non si possa dire con certezza, rimane comunque il fatto che le operazioni sono le medesime di quelle fatte dagli IP dannosi.
Si è visto che alcuni IP compaiono più di una volta, sia nello stesso file di log di un honeypot sia nei file di log di honeypot diversi. Nei casi di interazioni di breve durata vengono generate quasi sempre stesse eccezioni o errori, addirittura in alcuni casi anche la grandezza del pacchetto malformato generato nella risposta ha la stessa lunghezza in ogni interazione. Questo fa supporre che vengano utilizzati processi automatizzati al fine di scannerizzare la rete.
In altri casi invece il numero di comandi varia di numero e i comportamenti non sono esattamente uguali. Prendiamo in esame l' IP 172.105.246.XXX: nella tabella 5.9 si nota come la durata delle due sessioni sia diversa, addirittura nel primo caso viene generato un malfunzionamento di Conpot mentre nel secondo no. Nel caso dell' honeypot_3 viene causato di nuovo un blocco del servizio utilizzando questa volta una sequenza di eventi diversa. Per rispondere al punto (7), quindi, sembrerebbe che la connessione ripetuta in questi casi sia intenzionale e messa in atto manualmente. Quanto discusso avvalora inoltre la risposta al punto (4) riguardo alla lunghezza delle sessioni.
Probabilmente i dati raccolti non sono abbastanza per stilare una tecnica di *fingerprinting* vera e propria. Si é appreso però che gli IP malevoli operano in modo molto simile tra loro.
Un IP potrebbe essere bloccato se si verifica almeno una delle situazione di seguito:

- Richieste che generano errori di Mbap.
- Connessioni consecutive. Ad esempio, secondo quanto scritto sopra, se si è anche a conoscenza della localizzazione dell'IP e sapendo che almeno in due situazioni IP malevoli cinesi hanno tentato connessioni consecutive al device, un atteggiamento simile potrebbe far pensare ad operazioni illecite. Sarebbe opportuno limitare il numero di connessioni consecutive per esempio a due.
- Invio di dati non validi.
- Sessioni troppo lunghe. Quando si verificano casi del genere lo scopo dell'attaccante potrebbe essere quello di cercare punti deboli nel sistema. Evitare che

si verifichino sessioni di 6 minuti, ad esempio, potrebbe essere utile per celare delle vulnerabilità.

Gli eventi anomali di Conpot sembrano frutto di attacchi DoS, infatti ne è stato interrotto il normale funzionamento in diversi casi. Come è stato scritto precedentemente i device Modbus sono soggetti a questi tipi di attacchi e come viene sottolineato da Dodson, Beresford e Vingaard [14] possono verificarsi in modi diversi. Nel loro studio si sono verificati tre casi:

- Nel primo caso sono stati creati dei pacchetti specifici che hanno forzato il device a violare i loro vincoli in tempo reale. Si è trattato di un attacco che sfrutta piccoli pacchetti di traffico che usano poca banda, difficilmente distinguibili da pacchetti legittimi.

- Nel secondo caso l'attaccante ha preso di mira dispositivi con implementazioni incomplete dello stack del protocollo: nell'attacco vengono forniti comandi validi ma non implementati. Questo ha influenzato negativamente il controllo del processo del dispositivo.

- Nel terzo caso un buffer overflow ha compromesso la capacità di comunicazione di rete del dispositivo ma non ha influenzato il dispositivo stesso.

Considerando che durante la cattura pacchetti con Wireshark non si sono notati pacchetti particolari, è probabile che gli attacchi DoS verificatosi nel nostro studio rientrino nella tipologia del primo caso descritto.

### 6.0.1 Conclusioni

Conpot è uno strumento valido in ambito ICS ma deve essere affiancato ad altri tool. In una situazione reale con delle macchine di processo potrebbe essere utile far girare più istanze dello stesso per simulare un sistema più complesso, oppure potrebbe essere inglobato in un progetto più ampio (e.g, Gridpot). Conpot risulta però essere molto utile in una prima fase di studio dei comportamenti degli attaccanti in cui non si rischia di compromettere un sistema reale. Un' analisi iniziale è molto importante se non si hanno molti dati a disposizione o se si è in possesso di una conoscenza basilare delle casistiche che si possono verificare. Se si partisse infatti da ipotesi e si cercasse di implementare un sistema, si potrebbero inutilmente definire delle regole basandosi su casi che nella realtà si verificano raramente o addirittura mai.
Gli honeypot risultano essere degli ottimi alleati per comprendere e contrastare le interazioni malevole in ambito OT ma è necessario specializzarli maggiormente per operare nel campo industriale.
Un modo per renderli più efficienti potrebbe essere l'implementazione di una tecnica di *finger-printing* specifica per ICS, col fine di riconoscere e bloccare gli attaccanti. Questo risultato non è stato raggiunto in questa tesi per i pochi dati a disposizione:

sarebbe stato utile ottenere molte più interazioni diverse tra loro in modo da poter fare più assunzioni e confronti per appurare quanto detto nella risposta al punto (8) della sezione precedente.
Alla luce di quanto detto, stare al passo con la complessità degli attacchi risulta un compito assai arduo: un po' per la creatività degli attaccanti nello sviluppare nuovi metodi offensivi un po' per l'obsolescenza degli strumenti usati in ambito ICS. Per questo motivo, oltre agli honeypot e ad altri strumenti per la sicurezza, bisognerebbe avere anche una maggiore consapevolezza riguardo ai rischi che corrono questi sistemi.

### 6.0.2 Sviluppi Futuri

Il contributo che si vuole dare è, oltre all'approfondimento dei comportamenti degli attaccanti, quello di fornire dati ulteriori sulle tipologie di interazione con i dispositivi Modbus. In rete non è facile trovare dati di questo genere quindi potranno risultare utili per fare dei confronti o degli approfondimenti. Le analisi fatte potrebbero servire per affinare tecniche di *finger-printing* specifiche per gli attacchi ai dispositivi industriali Modbus.

# Bibliografia


[1] Graham Williamson. *"OT, ICS, SCADA – What's the difference?"*, 07 2015. https://tinyurl.com/y9xwnw2j.

[2] Tyson Macaulay and Bryan L Singer. *Cybersecurity for industrial control systems: SCADA, DCS, PLC, HMI, and SIS*. CRC Press, 2011.

[3] Hakan Kayan, Matthew Nunes, Omer Rana, Pete Burnap, and Charith Perera. Cybersecurity of industrial cyber-physical systems: A review. *ACM Comput. Surv.*, 54(11s), sep 2022.

[4] Keith Stouffer, Joe Falco, Karen Scarfone, et al. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[5] Sandep Bhandari. *"PLC vs RTU: differenza e confronto"*, 06 2023. http://tinyurl.com/56m8k88w.

[6] Jordan Moore. *"6 Key Terms in Upstream Oil and Gas Automation"*. http://tinyurl.com/49jj65k6.

[7] Charles J Brooks and Philip A Craig Jr. *Practical industrial cybersecurity: ICS, industry 4.0, and IIoT*. John Wiley & Sons, 2022.

[8] Francesco Santini. *"Come costruire e valutare un honeypot per sistemi industriali"*, 01 2022. https://tinyurl.com/bdhsmmpe.

[9] Trend Company. *"Industrial Control System"*. https://tinyurl.com/4pej23kk.

[10] Yikai Xu, Yi Yang, Tianran Li, Jiaqi Ju, and Qi Wang. Review on cyber vulnerabilities of communication protocols in industrial control systems. In *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–6, 2017.

[11] Eduard Paul Enoiu, Kejsi Biçoku, Cristina Seceleanu, and Michael Felderer. A taxonomy of vulnerabilities, attacks, and security solutions in industrial plcs. In *CyberSecurity in a DevOps Environment: From Requirements to Monitoring*, pages 3–33. Springer, 2023.

[12] Hung-Chang Chang, Chen-Yi Lin, Da-Jyun Liao, and Tung-Ming Koo. The modbus protocol vulnerability test in industrial control systems. In *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 375–378, 2020.

[13] Paulo Simoes, Tiago Cruz, Jorge Proença, and Edmundo Monteiro. On the use of honeypots for detecting cyber attacks on industrial control networks. 07 2013.
https://tinyurl.com/yt6knnuj.

[14] Michael Dodson, Alastair R. Beresford, and Mikael Vingaard. Using global honeypot networks to detect targeted ics attacks. In *2020 12th International Conference on Cyber Conflict (CyCon)*, volume 1300, pages 275–291, 2020.
https://tinyurl.com/mps8sx6v.

[15] Giorgio Sbaraglia. *"La Guerra Cibernetica: Stuxnet, il caso più famoso"*, 01 2020.
http://tinyurl.com/2p9bb9n8.

[16] Marie Baezner and Patrice Robin. Stuxnet. Technical report, ETH Zurich, 2017.

[17] Tim Ricketts. *"Sheep dip your removable storage devices to reduce the threat of cyber attacks"*.
http://tinyurl.com/3dkynexh.

[18] Kaspersky. *"ICS CERT di Kaspersky del primo semestre 2023"*, 09 2023.
http://tinyurl.com/4a3bkh3e.

[19] Andrea Cadei. *"Quali sono i principali protocolli industriali per il monitoraggio degli impianti?"*, 06 2019.
https://tinyurl.com/46p24hub.

[20] Jeffrey Hibbard. *"5 Real-Time, Ethernet-Based Fieldbuses Compared"*, 05 2016.
https://tinyurl.com/3b6yct78.

[21] P. Brooks. Ethernet/ip-industrial protocol. In *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597)*, volume 2, pages 505–514 vol.2, 2001.
https://tinyurl.com/yc7zhpf6.

[22] Harald Kleines, Sebastian Detert, Matthias Drochner, and Frank Suxdorf. Performance aspects of profinet io. *IEEE Transactions on Nuclear Science*, 55(1):290–294, 2008.
https://tinyurl.com/yfrz865h.

[23] Ankush Meshram. *Self-learning Anomaly Detection in Industrial Production*. KIT Scientific Publishing, 2023.

[24] Ben Edrington, Bingyan Zhao, Adam Hansel, Masahiko Mori, and Makoto Fujishima. Machine monitoring system based on mtconnect technology. *Procedia Cirp*, 22:92–97, 2014.
https://tinyurl.com/25hcasm3.

[25] Stefan-Helmut Leitner and Wolfgang Mahnke. Opc ua–service-oriented architecture for industrial applications. 2006.
https://tinyurl.com/4eventuh.

[26] Andy Swales et al. Open modbus/tcp specification. *Schneider Electric*, 29(3):19, 1999.
https://tinyurl.com/bdww9fa8.

[27] Massimo Giussani. *"Automazione oggi tutorial"*, 11 2006.
http://tinyurl.com/3mf9d44h.

[28] Modbus organization. *"Modbus application protocol specification"*.
http://tinyurl.com/3p9kyduc.

[29] Stefano Di Paola. *"Testing the Security of Modbus Services"*, 03 2024.
https://tinyurl.com/44dwrefy.

[30] Jschulze. *"Question about ModBus slave addresses"*, 09 2022.
http://tinyurl.com/ynt6jpwc.

[31] Modbus organization. *"Modbus tools"*.
https://tinyurl.com/jw78j3ex.

[32] Modbus organization. *"Modbus tools"*.
https://tinyurl.com/y3nphfpy.

[33] *"PLCynergy"*, 01 2021.
https://plcynergy.com/Modbus-tcp-protocol/.

[34] Omar Morando. *"Blog di Omar Morando"*.
https://blog.omarmorando.com/.

[35] *"EverElettronica"*.
http://tinyurl.com/2wujaesz.

[36] Sebastien Tricaud. *"What Deploying Honeypots to Catch ICS Attacks Taught Me"*, 12 2018.
https://tinyurl.com/y5np32hv.

[37] J.Riden C.Seifert. *"A Guide to Different Kinds of Honeypots"*, 11 2010.
https://tinyurl.com/mrynb6nj.

[38] Lance Spitzner. *Honeypots: tracking hackers*, volume 1. Addison-Wesley Reading, 2003.

[39] Niels Provos et al. A virtual honeypot framework. In *USENIX Security Symposium*, volume 173, pages 1–14, 2004.
https://tinyurl.com/33cwsk2h.

[40] Vito Lavecchia. *"Progetto Honeyd: creazione di un sistema honeypot"*.
https://vitolavecchia.altervista.org/progetto-honeyd\
-creazione-un-sistema-honeypot/.

[41] Niels Provos. Honeyd-a virtual honeypot daemon. In *10th dfn-cert workshop, hamburg, germany*, volume 2, page 4, 2003.
https://tinyurl.com/2etcbknz.

[42] Radek Hes, Peter Komisarczuk, Ramon Steenson, and Christian Seifert. The capture-hpc client architecture. *Technical report, Victoria University of Wellington*, 2009.
https://tinyurl.com/nhkf5j4e.

[43] Efrén López-Morales, Carlos Rubio-Medrano, Adam Doupé, Yan Shoshitaishvili, Ruoyu Wang, Tiffany Bao, and Gail-Joon Ahn. Honeyplc: A next-generation honeypot for industrial control systems. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 279–291, 2020.
https://github.com/sefcom/honeyplc.

[44] Incibe-cert. *"Honeypot, a tool to know your enemy"*, 06 2018.
https://tinyurl.com/29b85y2n.

[45] Jeffrey T Dougherty. *Evasion of honeypot detection mechanisms through improved interactivity of ICS-based systems*. PhD thesis, Monterey, CA; Naval Postgraduate School, 2020.
https://tinyurl.com/j2cjp8hz.

[46] Arthur Jicha, Mark Patton, and Hsinchun Chen. Scada honeypots: An in-depth analysis of conpot. In *2016 IEEE conference on intelligence and security informatics (ISI)*, pages 196–198. IEEE, 2016.
http://tinyurl.com/ntd5rs9v.

[47] MushMush Foundation. *"Conpot"*, 2014.
http://conpot.org/.

[48] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. Gotta catch'em all: a multistage framework for honeypot fingerprinting. *Digital Threats: Research and Practice*, 4(3):1–28, 2023.
https://dl.acm.org/doi/pdf/10.1145/3584976.

[49] Christian Feuchter. *"Luring the Threat: Lessons from ICS Honeypots in Ukraine and Germany"*, 02 2024.
https://tinyurl.com/4hvnpujt.

[50] Dahae Hyun. *Collecting cyberattack data for industrial control systems using honeypots*. PhD thesis, Monterey, California: Naval Postgraduate School, 2018. https://tinyurl.com/5d67fr23.