

Reverse Engineering

per dispositivi IoT

MI OCCUPO DI

- Informatica forense
- Sviluppo software
- RecuperaBit, Carbon14, ...



PERCORSO

- IoT e schede a relay
- Reversing di un'app Android
- Implementazione in Python

INTERNET OF THINGS

Il termine “Internet delle cose” indica l’estensione della rete tramite il **collegamento di oggetti concreti**. Questi possono essere situati all’interno di un’abitazione (domotica).



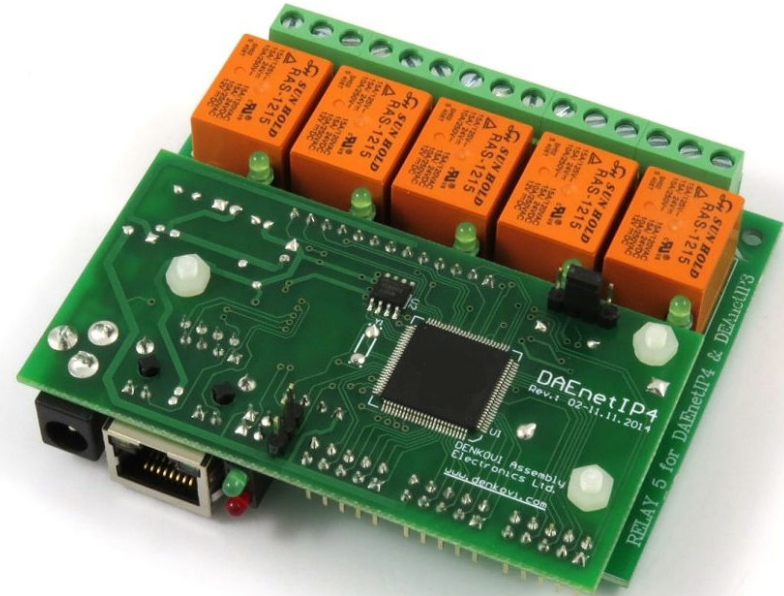
CONNETTO

GIRO, VEDO GENTE, MI MUOVO, CONOSCO, FACCIO DELLE COSE

SCHEDE A RELAY

Possibili componenti:

- Sensori (input)
- Attuatori (relay, PWM)
- Sistema di controllo (collegato alla rete)

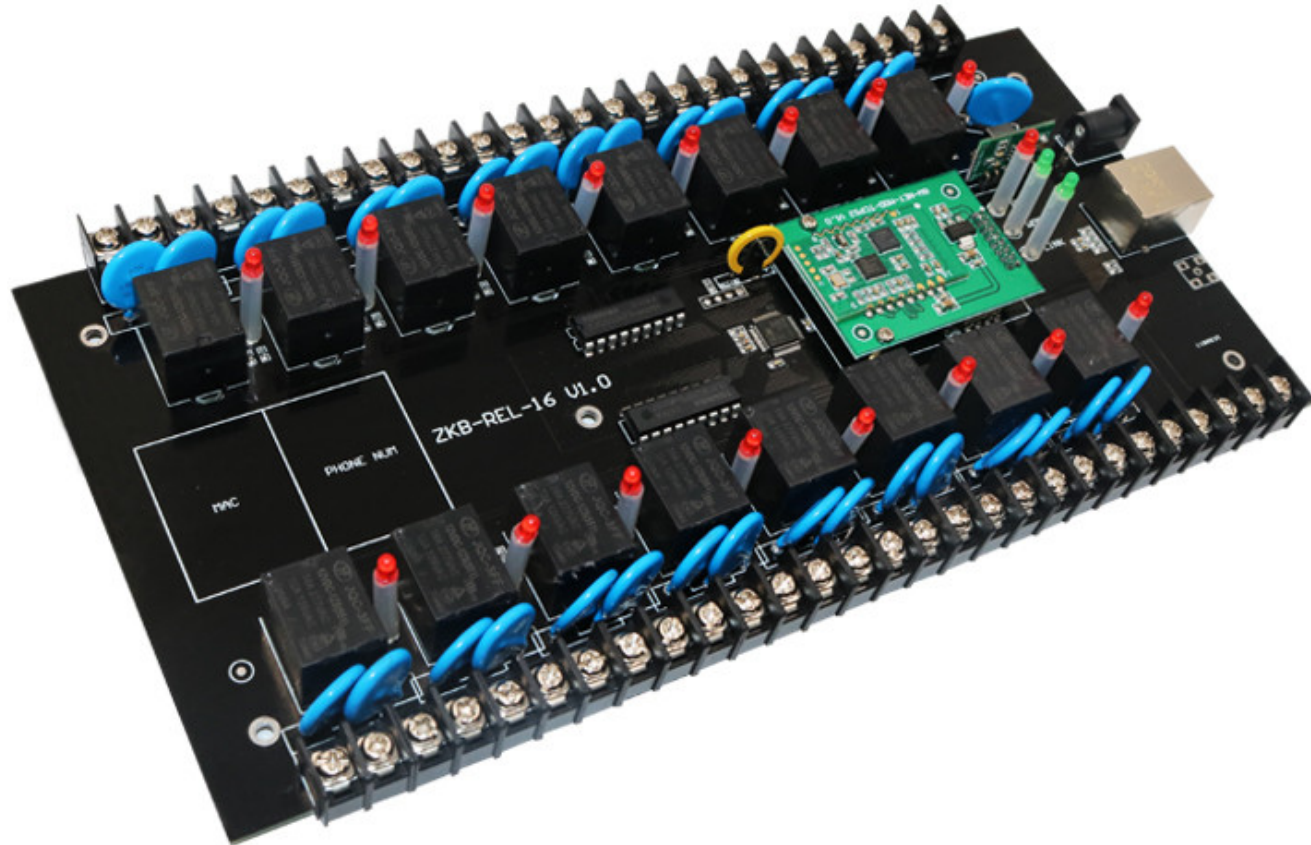


CARATTERISTICHE VARIABILI

Possono avere:

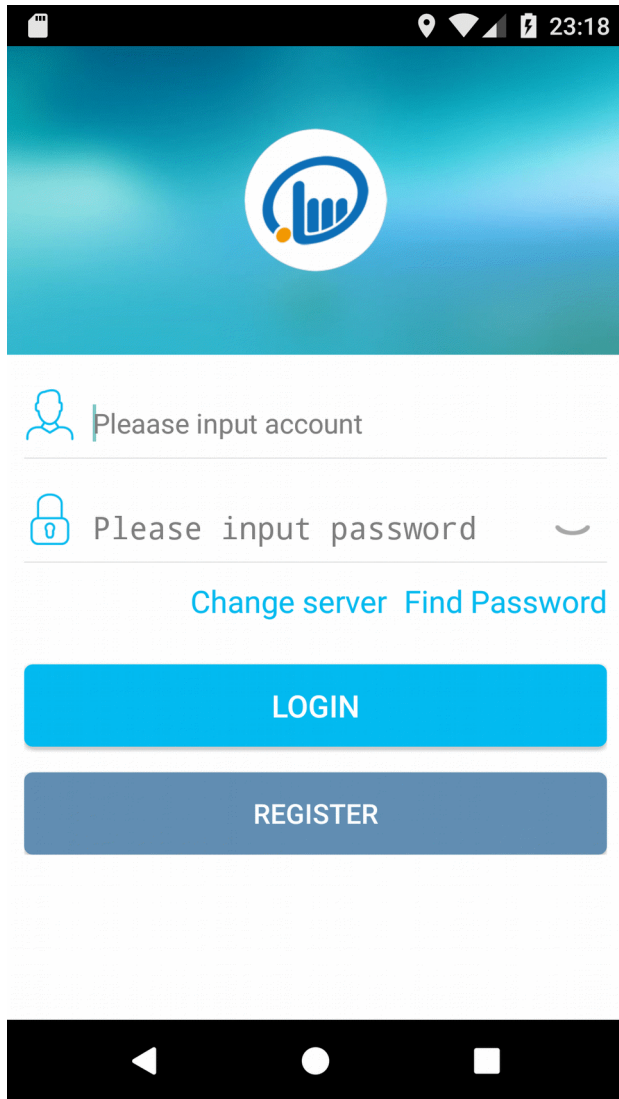
- Interfaccia web
- Password
- Costo contenuto
- Documentazione decente

CASE STUDY: ZKB-REL-16



CARATTERISTICHE

- Collegamento Ethernet
- Applicazione Android di dubbia provenienza,
con login obbligatorio



Please input account



No waay!

MANUALE (DELLA VERSIONE WI-FI)



www.zhangkongbao.com

目录

ZKB-REL-16-W 使用手册.....	1
目录	2
1 初步设置和测试.....	4
1.1 下载掌控宝软件.....	4
1.1.1 手机端.....	4
1.1.2 电脑端.....	4
1.2 初步设置（以安卓手机为例）.....	5
1.2.1 APP 安装	5

TRADUCENDO...



www.zhangkongbao.com

table of Contents

ZKB-REL-16-W Manual 1

table of Contents..... 2

1 initial set up and test 4

 Treasure 1.1 download control software 4

 1.1.1 mobile terminal 4

 1.1.2 PC side 4

 1.2 preliminary settings (to Android phones, for example) 5

 1.2.1 APP install 5

INFORMAZIONI ACQUISITE

- La app Android può controllare direttamente i dispositivi in LAN
- Controllo tramite MODBUS (non chiaro)
- La password è **admin**... che domande!

“

*The S in IoT stands
for Security*

INTERFACCIA WEB (1)

firmware revision : v4015 中文



Current Status	parameter	Help
Local IP Config	Module Name : ZKB-IOT-RELAY	• Current IP Address : default IP of module • Remote IP/TX /RX : IP of server or device connecting with module;reset for disconnect TX/RX:data volume that each server or device communicates with module;reset for disconnect • TX Count/RX Count : a total of data volume that servers or devices communicate with module;reset for power off
Serial Port	Current IP Address : 10.42.0.106	
Misc Config	MAC Address : d8-b0-4c-b7-ca-0d	
Reboot	SocketA Status : connected	
	SocketB Status : Remote IP / TX / RX	
	-1 : 0.0.0.0 / 0 byte / 0 byte	
	-2 : 0.0.0.0 / 0 byte / 0 byte	
	-3 : 0.0.0.0 / 0 byte / 0 byte	
	-4 : 0.0.0.0 / 0 byte / 0 byte	
	-5 : 0.0.0.0 / 0 byte / 0 byte	
	SocketB TX Count/RX Count : 8/ 51 bytes	

firmware revision : v4015 中文



Current Status	parameter	Help
Local IP Config	IP type : DHCP	• IP type: StaticIP or DHCP • StaticIP: Module's static ip • Submask: usually 255.255.255.0 • Gateway: Usually router's ip address • DNS IP: DNS gateway or Router's IP
Serial Port	Static IP : 192 . 168 . 0 . 7	
Misc Config	Submask : 255 . 255 . 255 . 0	
Reboot	Gateway : 192 . 168 . 0 . 1	
	DNS Server : 208 . 67 . 222 . 222	
	Save Cancel	

INTERFACCIA WEB (2)

firmware revision : v4015 中文



Current Status	parameter	Help
Local IP Config	Local Port Number : 0 (1~65535)	
Serial Port	Remote Port Number : 8899 (1~65535)	
Misc Config	Work Mode : TCP Client	
Reboot	Remote Server Addr : www.zhangkongbao.com	
	Heartbeat Packet Type : None ASCII	
	Register Packet Type : Enable lonhand	
	TCP Server-kick off old connection : ☐	
	SocketB NTML : ☒	
	Save Cancel	

- **Custom Heartbeat Packet :**
this function is not open,not do support the Chinese , and 40 bytes in length
- **Custom Register Packet :**
this function is not open,not do support the Chinese , and 40 bytes in length
- **TCP Server-kick off old connection :**
TCP Server mode, a new connection whether to kick off the old connection,

firmware revision : v4015 中文

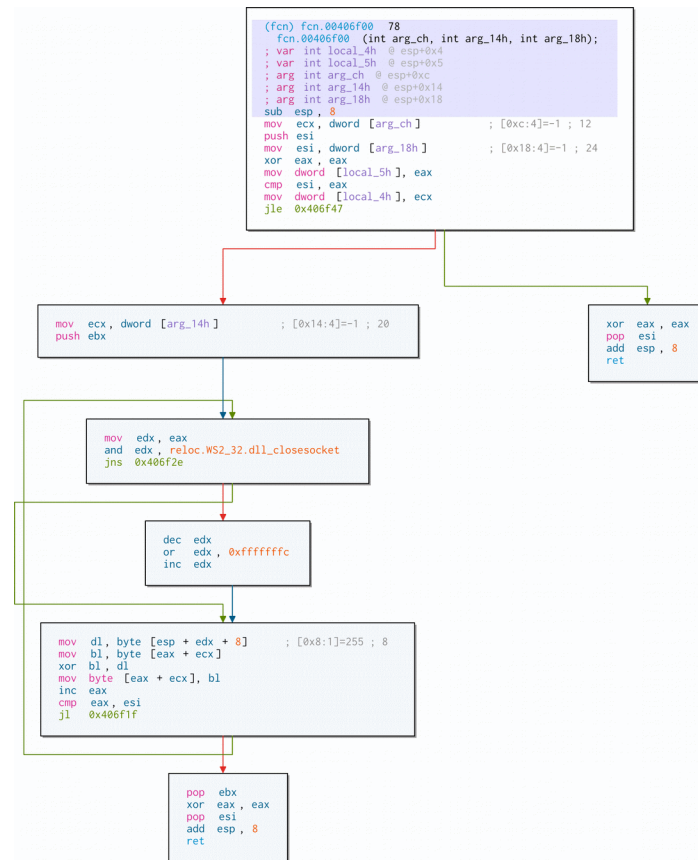


Current Status	parameter	Help
Local IP Config	Module Name : ZKB-IOT-RELAY	
Serial Port	Webserver Port : 80	
Misc Config	Username : admin	
Reboot	Password : admin	
	Reset Timeout : 3600 (s)(60~8640000)	
	Save Cancel	

- **Timeout Restart Time:**
When the network port or serial port without data, timeout restart.0s,Close the function

REVERSE ENGINEERING

- Studio di un software chiuso, per riuscire a capire cosa fa
- *Normalmente* richiede di disassemblare un file eseguibile



NEL CASO DI ANDROID

- Le app Android sono basate su Java
- Il bytecode è facile da decompilare
- Installiamo MobSF tramite Docker



Drag files here or click Upload & Analyze

 Upload & Analyze

Search MD5

[Recent Scans](#) | [API Docs](#) | [About](#)

- Information
- Scan Options
- Signer Certificate
- Permissions
- Binary Analysis
- Android API
- Browsable Activities
- Security Analysis
- Malware Analysis
- Reconnaissance
- Components
- Download Report
- Start Dynamic Analysis

Icon



File Information

Name LonHand_V3.0.170422.apk
Size 6.07MB
MD5 9883d7efff0bc7d04e5d76c7db5d50b7
SHA1 04fdf278931c8fa07b5635ab0b2cc92a3c598e7b
SHA256 b86e80b837fd6bab5a656b42197a65bb05789ca12cff0363b83587daf3659e3f

App Information

Package Name com.usr.newiot
Main Activity .LoginActivity
Target SDK 14 **Min SDK** 14 **Max SDK**
Android Version Name 3.0.170422
Android Version Code 301700422

31
ACTIVITIES

[View](#)

EXPORTED
ACTIVITIES
0

1
SERVICES

[View](#)

EXPORTED
SERVICES
1

4
RECEIVERS

[View](#)

EXPORTED
RECEIVERS
4

0
PROVIDERS

[View](#)

EXPORTED
PROVIDERS
0

Scan Options

[Rescan](#)[Start Dynamic Analysis](#)

View Code

[View Java](#)[Download Java Code](#)[View Smali](#)[Download Smali Code](#)[View AndroidManifest.xml](#)

Signer Certificate

```
[
[
  Version: V3
  Subject: CN=LonHand, OU=??, O=????, L=JiNan, ST=ShanDong, C=CN
  Signature: SHA-256withRSA, DN: cn=lonhand, ou=lonhand, o=lonhand, c=cn
]
```

ANALISI DEL CODICE

- Verificare **strings.xml** per potenziali URL
- Ignorare le directory di librerie standard (Google, Firebase, Facebook, eccetera)
- Nel nostro caso il codice del *vendor* è in:

root/.MobSF/uploads/9883d7efff0bc7d04e5d76c7db5d50b7/java_source/
com/usr/iotcommunication

STRUMENTI UTILI

- **Atom** o **VS Code** — lettura del codice
- **Angry IP Scanner** — ricerca del dispositivo
- **nmap** — scansione delle porte
- **ag** — ricerca testuale efficiente

PASSO PER PASSO

- Diamo `nmap -P0 -p- 192.168.8.176` e rileviamo tre porte aperte: 80, 1501, 8899
- Troviamo la porta 8899 nella classe **Device**:

```
> ag -C 2 8899
com/usr/newiot/bean/Device.java
29-     private String name;
30-     private String pasd = "admin";
31:     private int port = 8899;
32-     private int state = 0;
33-     private int type;
```

- Cercando `new Device` scopriamo `com/usr/newiot/IotApplication.java` con dentro:

```
public void checkLocalDeviceState(final Device device) {  
    [...]  
    if (IotApplication.this.iotComm.connectDeviceWithMAC([...]  
        device.getPasd(), device.getIp(), device.getPort(), 500))  
    [...]  
}
```


- La definizione di `connectDeviceWithMAC` si trova in `IOTComm.java`:

```
public boolean connectDeviceWithMAC(...) {  
    object2 = [...] ? new LocalConnect(...) : [...]
```

- Dentro a `LocalConnect.java`:

```
Socket socket = this.socket;  
[...]  
this.send((String.valueOf(this.pasd) + "\r\n").getBytes("utf-8"));
```

- Possiamo provare la password tramite `telnet`

- Cercando **.send()** troviamo il metodo **sendData** dentro a **IOTComm.java**
- Viene usato in molti posti:

```
> ag '\.sendData\('
com/usr/newiot/util/CmdUtil.java
[...]
497:         iotApplication.sendData(string2, CmdUtil.getAllInState());
501:         iotApplication.sendData(string2, CmdUtil.getAllOutState());
[...]
```

- Dentro a **CmdUtil.java** troviamo **finalmente** i comandi per controllare i led:

```
public static byte[] getCtrlOutCloseCmd(int n2) {  
    return StringUtil.generateCmd(new byte[]{0, 1, (byte)(n2 & 255)});  
}
```

```
public static byte[] getCtrlOutOpenCmd(int n2) {  
    return StringUtil.generateCmd(new byte[]{0, 2, (byte)(n2 & 255)});  
}
```

PIANO D'AZIONE

- Replichiamo in Python la funzione che genera le stringhe dei comandi
- Creiamo una piccola classe per il dispositivo
- Testiamo tutto!

TRADUZIONE IN PYTHON

```
public static byte[] generateCmd(byte[] arrby) {
    int n2 = arrby.length + 5;
    byte[] arrby2 = new byte[n2];
    arrby2[0] = 85;
    arrby2[1] = -86;
    byte[] arrby3 = StringUtil.int2byte(arrby.length);
    arrby2[2] = arrby3[1];
    arrby2[3] = arrby3[0];
    arrby2[n2 - 1] = (byte)(arrby2[2] + arrby2[3]);
    int n3 = 0;
    while (n3 < arrby.length) {
        arrby2[n3 + 4] = arrby[n3];
        int n4 = n2 - 1;
        arrby2[n4] = (byte)(arrby2[n4] + arrby[n3]);
        ++n3;
    }
    return arrby2;
}
```

```
def _generateCmd(arrby):
    n2 = len(arrby) + 5
    arrby2 = list('\x00' * n2)
    arrby2[0] = chr(85)
    arrby2[1] = chr(-86%256)
    arrby3 = struct.pack('<I', len(arrby))
    arrby2[2] = arrby3[1]
    arrby2[3] = arrby3[0]
    arrby2[n2 - 1] = chr(ord(arrby2[2]) + ord(arrby2[3]))
    n3 = 0
    while (n3 < len(arrby)):
        arrby2[n3 + 4] = chr(arrby[n3])
        n4 = n2 - 1
        arrby2[n4] = chr(ord(arrby2[n4]) + arrby[n3])
        n3 += 1
    return ''.join(arrby2)
```

LETTURA DELLO STATO DEI RELAY

- Per leggere lo stato bisogna richiamare **generateCmd** con parametro **[0, 10]**
- Verifichiamo la risposta quando i led accesi sono 3, 4, 5, 12, 14 (da 1 a 16):

```
> python prova.py  
aa550004008a1c28d2  
['0b10101010', '0b1010101', '0b0', '0b100', '0b0', '0b10001010',  
'0b11100', '0b101000', '0b11010010']
```

CLASSE FINALE

```
class Board(object):
    def __init__(self, address, port, password):
        self.address = address
        self.port = port
        self.password = password

    def connect(self):
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.connect((self.address, self.port))
        self.socket.send('{}\r\n'.format(self.password))
        self.socket.recv(1024)    # Flush!
        time.sleep(0.125)

    def close(self):
        self.socket.close()

    @staticmethod
    def _generateCmd(arrby):
        [...]
```

```
    def trigger(self, output, on = True):
        self.connect()
        action = 2 if on else 1
        self.socket.send(self._generateCmd([0, action, output])+'\r\n')
        self.socket.recv(1024)    # Flush!
        self.close()

    def status(self):
        self.connect()
        self.socket.send(self._generateCmd([0, 10]) + '\r\n')
        result = self.socket.recv(1024)[6:8]
        self.close()
        lower = ord(result[0])
        upper = ord(result[1])
        merged = lower + upper*256
        result = {}
        for i in range(1, 17):
            result[i] = bool(merged / (2**(i-1)) % 2)
        return result
```

ATTENZIONE!

- Dopo l'invio della password tramite socket è necessaria **una breve pausa**
- Bisogna sempre effettuare una **recv** se il dispositivo risponde con una stringa
- Apriamo il socket ogni volta, per evitare inceppamenti



IDEE PER L'AUTOMAZIONE

- Controllare l'innaffiamento del giardino
- Temporizzare le persiane
- Accendere le luci da smartphone