

# **Bezpieczeństwo systemów i sieci**

dr inż. Krzysztof Cabaj

# Plan wykładu

Sprawy organizacyjne

Wykład wprowadzający

# O mnie ...

dr inż. Krzysztof Cabaj  
kcabaj@elka.pw.edu.pl

Prowadzę zajęcia laboratoryjne z BSS od pierwszej edycji przedmiotu z ćwiczeniami

Przez wiele lat byłem instruktorem certyfikowanych kursów Cisco: CCNA, CCNP, CCNA Security (dawniej Network Security )

Biorę udział w audytach i analizie po włamaniowej

Jestem współopiekunem Koła Naukowego Bezpieczeństwa Informatyki (KNBI)



# Zaliczenie

Laboratorium 7 ćwiczeń ( $2 \times 5p + 5 \times 6$ ) = 40p

Egzamin 60p

Warunki konieczne zaliczenia

- Co najmniej 21p z laboratorium
- Co najmniej 31p z egzaminu

Ocena (51-60p – 3, 61 – 70p - 3.5 .... 90-100p - 5)

# Przedmioty na wydziale o podobnej tematyce

PTKB Podstawy teoretyczne kryptografii i ochrony informacji  
prof. dr hab. Tomasz Adamski

MKOI Zaawansowane metody kryptografii i ochrony informacji  
prof. dr hab. Zbigniew Kotulski

PKRY Protokoły kryptograficzne  
prof. dr hab. Zbigniew Kotulski

ISL Informatyka śledcza  
dr inż. Magdalena Szeżyńska

BEST Bezpieczeństwo sieci teleinformatycznych  
dr inż. Wojciech Mazurczyk

BOT Bezpieczeństwo oprogramowania i testy penetracyjne  
prof. nzw. dr hab. Krzysztof Szczypiorski

# Plan wykładu

1. Wprowadzenie
2. Szyfry symetryczne/asymetryczne
3. Funkcje skrótu - uwierzytelnienie
4. Infrastruktura klucza publicznego
5. Wprowadzenie do zagrożeń sieciowych
6. Bezpieczeństwo aplikacji C/C++
7. Karty inteligentne (zaproszony dr inż. P.Nazimek)
8. Wirusy, robaki, konie trojańskie (malware I)
9. Wirusy, robaki, konie trojańskie (malware II)
10. Bezpieczeństwo aplikacji Webowych
11. Systemy zapewniających bezpieczeństwo w sieciach komputerowych
12. Mechanizmy bezpieczeństwa systemów operacyjnych + VPN
13. Mechanizmy monitorowania i logowania systemów operacyjnych
14. Systemy HoneyPot
15. Polityka bezpieczeństwa

# Laboratorium

Ćwiczenie 1. Algorytmy szyfrowania 1	5p
Ćwiczenie 2. <i>GPG</i>	6p
Ćwiczenie 3. Openssl	6p
Ćwiczenie 4. Stunnel	5p
Ćwiczenie 5. Systemy IDS	6p
Ćwiczenie 6. Bezpieczeństwo aplikacji	6p
Ćwiczenie 7. Bezpieczeństwo aplikacji Webowych	6p

# Literatura

Inżynieria zabezpieczeń, Ross Anderson

Kryptografia dla praktyków, Bruce Schneier

Kryptografia stosowana, Alfred Menezes, i inni

Internet, osobiście polecam (czytuję codziennie)  
witrynę

<http://www.dshield.org>



Czy mają Państwo pytania  
do części organizacyjnej ?

# Wykład wprowadzający

## Bezpieczeństwo IoT

# Pokaz

Automatyzacja bramy garażowej w nowoczesny i modny dzisiaj sposób – z wykorzystaniem sprzętu IoT.

Podpięcie urządzenia otwierającego bramę do sieci i możliwość sterowania za pomocą komórki, laptopa ...

# MyDoorOpener

[MyDoorOpener.com](#)

FeaturesAssemblyConfigDownloadsSupport

## Welcome to MyDoorOpener.com

MyDoorOpener is a native iPhone App, combined with inexpensive and easy to setup hardware, that allows your garage door to be securely opened, closed or monitored, directly from the palm of your hand. Other types of devices can also be controlled and monitored, only limited by your imagination.

### News

- [Versions 3.2](#) is now available with great new features!
- New [simplified assembly process](#) is now available.

Available on the App Store

[Tweet](#) [Follow @yanavery](#)





### Features

What features are supported by MyDoorOpener.



### Assembly

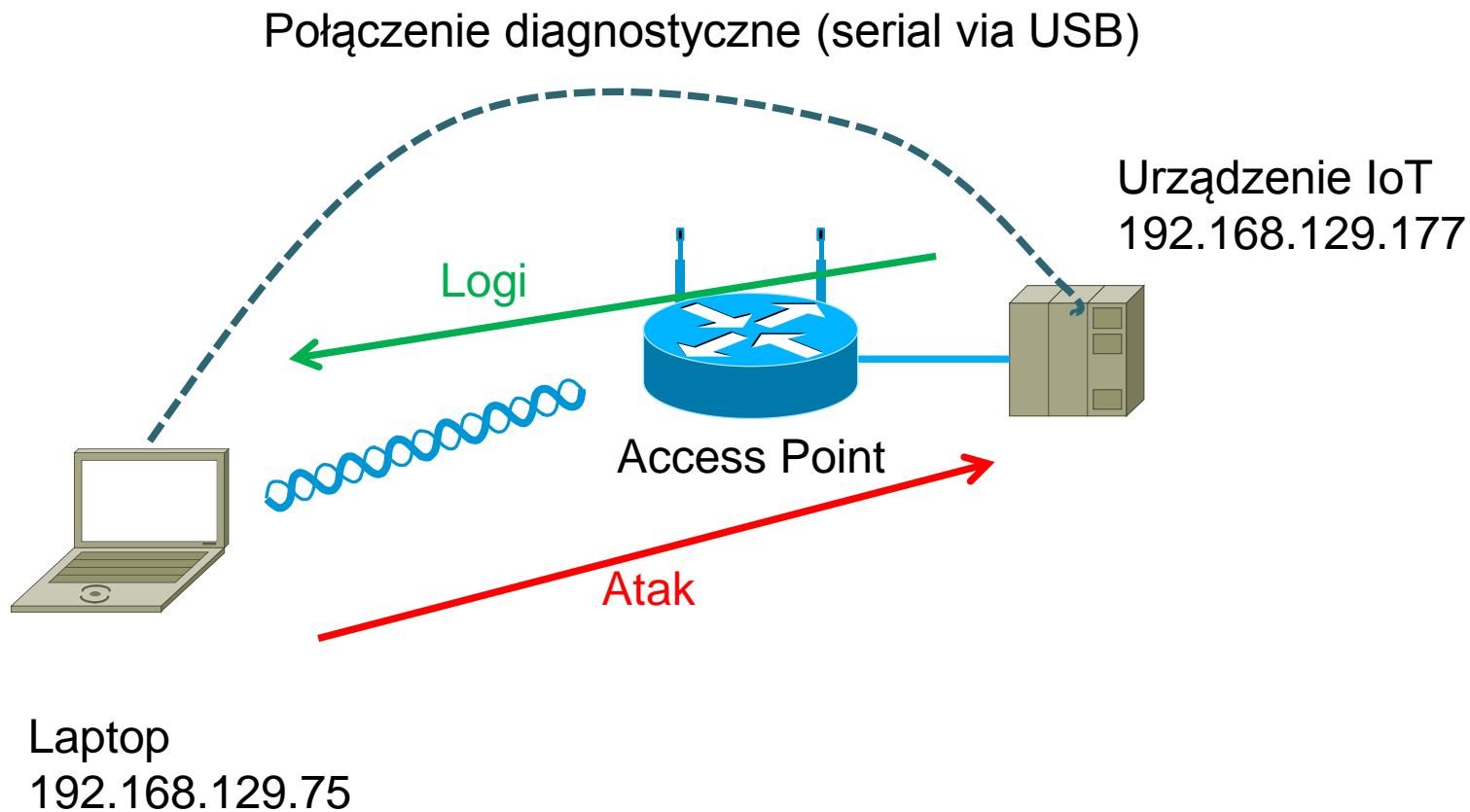
How to build and assemble the hardware.



### Config

How to configure the software.

# Schemat połączenia urządzeń



# Bezpieczeństwo

A jak wygląda sprawa bezpieczeństwa tego typu urządzeń

# Podstawowe zasady bezpieczeństwa

... dla „zwykłych” komputerów

- update-y oprogramowania
- szyfrowanie transmisji
- silne hasła
- systemy AV
- zapory ogniowe
- . . .



# Czy patrząc na takie urządzenie ...



... myślimy o komputerze ?



# A „komputery” mogą być jeszcze mniejsze



## Intel Edison

Developer	Intel Corporation
Type	Tiny Board Computer
Release date	Q3'14
Discontinued	June 19, 2017
CPU	Atom 2-Core (Silvermont) @ 500 MHz
Memory	(LPDDR3 1 GB)
Storage	4 GB EMMC

# Jak najczęściej zbudowane są urządzenia IoT

## Budowa systemu IoT

- System komputerowy (hardware)
- System operacyjny (najczęściej Linux)
- Interfejs użytkownika (najczęściej poprzez przeglądarkę)

## Co może pójść nie tak

- Brak update-ów oprogramowania
- Błędy w dedykowanym oprogramowaniu

# Wykryte błędy w urządzeniach IoT

- Brak szyfrowania (dostęp poprzez telnet, http)
- Słabe hasła (root/12345)
- Błędy w oprogramowaniu (ShellShock na urządzeniach QNAP)
- Błędy w interfejsie webowym
- Botnet Mirai wykorzystywał do ataków DDoS kamery IP oraz urządzeń typu DVR
- . . .

# Shodan

Jest usługą w Internecie, która cyklicznie skanuje wybrane porty na maszynach podpiętych do Internetu i zbiera do dalszych analiz uzyskane odpowiedzi (ang. bannery)

Serwis udostępnia możliwość przeszukiwania uzyskanych wyników, wyszukiwania określonych fraz oraz filtrowanie, przykładowo po kraju

# Shodan

Po doświadczeniach z Webduino przeszkukałem bazę Shodana w celu wyszukania bannerów zawierających frazę „webduino”

The screenshot shows the Shodan search interface. At the top, the search bar contains 'webduino' and the results count is 188. Below this, a world map highlights countries with red dots, and a table lists the top countries by result count. On the right, two search results are displayed, each showing the IP address, host information, and a list of HTTP banners. The first result is for IP 163.131.211.186, identified as 'Shonan Cable Network' in Japan, with banners including 'HTTP/1.0 200 OK' and 'Server: Webduino/1.7'. The second result is for IP 173.239.88.171, identified as 'Cascade Access, LLC' in the United States, with banners including 'HTTP/1.0 200 OK', 'Server: Webduino/1.7', and an XML response.

**SHODAN** webduino [Q](#) [Explore](#) [Developer Pricing](#) [Enterprise Access](#) [Contact Us](#)

[Exploits](#) [Maps](#)

**TOTAL RESULTS**  
188

**TOP COUNTRIES**

Country	Count
United States	37
Italy	36
Germany	18
Taiwan	14
United Kingdom	11

**Front Page**  
163.131.211.186  
g163-131-211-186.scn-net.ne.jp  
**Shonan Cable Network**  
Added on 2018-05-08 15:13:43 GMT  
• Japan, Oiso  
Technologies: **Angular Material**   
[Details](#)

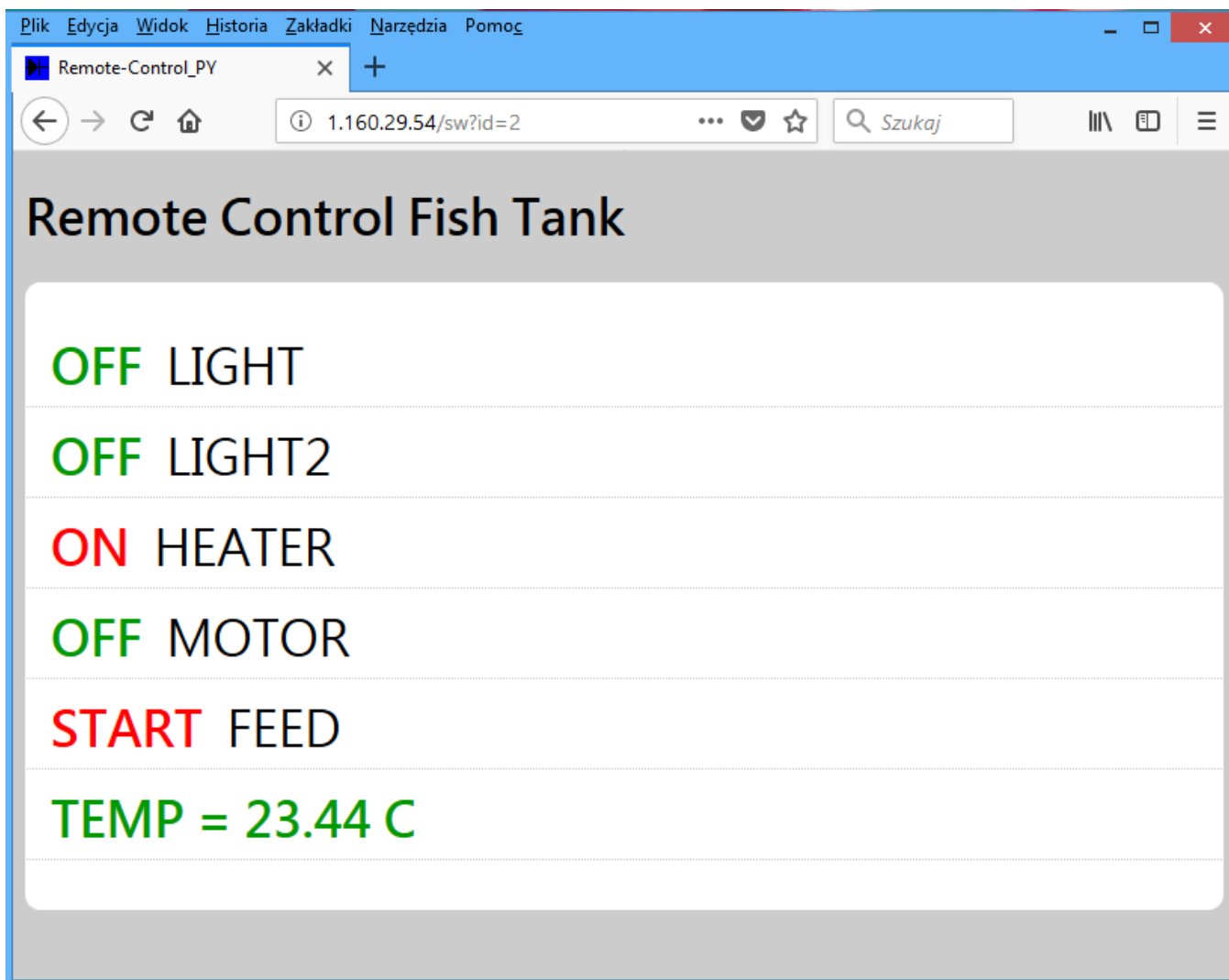
**173.239.88.171**  
pppoe-173-239-88-171.riocaccess.com  
**Cascade Access, LLC**  
Added on 2018-05-08 14:49:31 GMT  
 United States, Mesquite  
[Details](#)

HTTP/1.0 200 OK  
Server: **Webduino/1.7**  
Access-Control-Allow-Origin: \*  
Content-Type: text/html; charset=utf-8

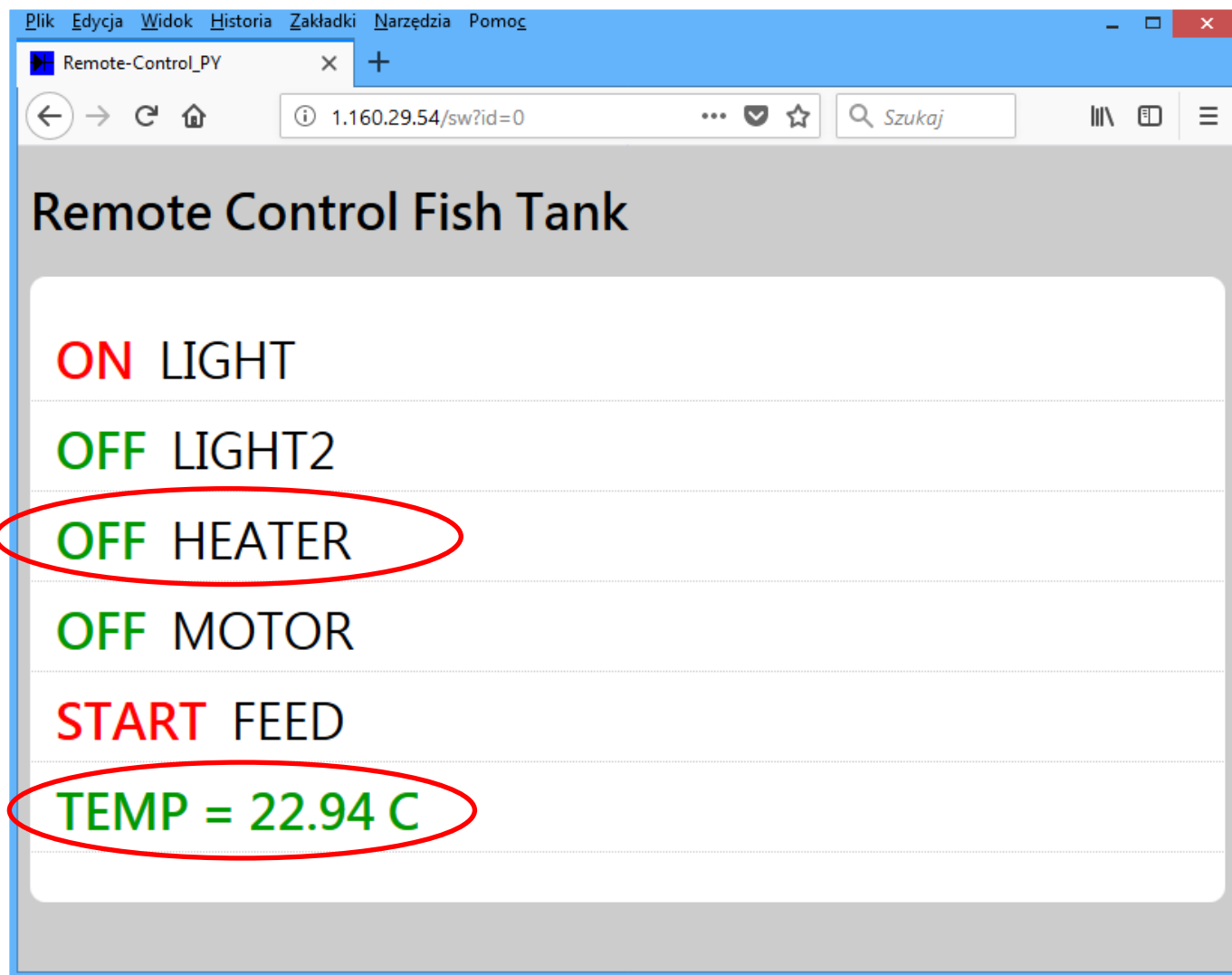
HTTP/1.0 200 OK  
Server: **Webduino/1.7**  
Access-Control-Allow-Origin: \*  
Content-Type: text/xml; charset=utf-8

```
<?xml version="1.0"?>
<myDoorOpener>
<status statusPin="3">Opened</status>
```

# Shodan wyniki



# Shodan wyniki



# Shodan wyniki

Web browser window showing the "地中海太陽能溫控系統" (Mediterranean Solar Temperature Control System) interface. The interface displays temperature data and system status.

**Header:** 地中海 太陽能 回家洗澡真好 0800-234511

**Temperature Data:**

- 保溫桶 (Insulation Tank):
  - 3. 上段水溫 (Upper Water Temperature): **29.9 °C**
  - 2. 中段水溫 (Middle Water Temperature): **29.5 °C**
- 副桶 (Sub-Tank):
  - 5. 出口水溫 (Outlet Water Temperature): **15 °C**
- 溫控: **53 °C**

**System Status:** 目前太陽能定時加熱已關閉OFF

**Time:** 11:55

**Buttons:** 強制ON, —, 設定, 我的地中海





# Shodan wyniki


← → ↺ 🏠 ⓘ 185.15.171.46:8080

## My Heating


TEMPERATURA DESIDERATA

21.00°  

TEMPERATURA RILEVATA

19.94° 

ACCENSIONE MANUALE

STATUS: **OFF**  **GAS**

[ATTIVA](#) | [DISATTIVA](#)

# Shodan wyniki



# Shodan wyniki

Często wśród wyników pojawiała się odpowiedź w XML-u postaci

```
<?xml version="1.0"?>  
<myDoorOpener>  
  <status statusPin="3">Opened</status>  
  <challengeToken>Cyber134429</challengeToken>  
</myDoorOpener>
```

O tych urządzeniach więcej za chwilę . . .

# Czy ktoś podłącza systemy sterowania do Internetu?

Dane z systemu Shodan z 2018.02.17

Modbus	8 740	PCWorks	487
Siemens S7	2 013	Melsec-Q (Mitsubishi)	71
DNP3	255	OMRON FINS	866
Tridium	19 228	RedLion (HMI) Crimson v3	690
BACnet -	10 458	Codesys	1738
EtherNet/IP	18 488	IEC 60870-5-104	369
GE-SRTP	50	ProConOS	118
HART-IP (Fieldbus)	12		

W sumie 63 583 systemów !

# Shodan wyniki

Więcej ciekawych (naprawdę dużych systemów) wyszukanych poprzez Shodan-a zawiera prezentacja

**„System Shock: The Shodan computer search engine”, Dan Tentler**

# MyDoorOpener

Pojawiający się często XML jest odpowiedzią od urządzenia Do-it-Yourself służącego do otwierania bramy garażowej

Podczas badań wykryto kilkadziesiąt takich urządzeń podpiętych do Internetu

Przykładowo 2018.04.20 działały co najmniej 23 takie urządzenia

# MyDoorOpener

[MyDoorOpener.com](#)[Features](#)[Assembly](#)[Config](#)[Downloads](#)[Support](#)

## Welcome to MyDoorOpener.com

MyDoorOpener is a native iPhone App, combined with inexpensive and easy to setup hardware, that allows your garage door to be securely opened, closed or monitored, directly from the palm of your hand. Other types of devices can also be controlled and monitored, only limited by your imagination.

### News

- [Versions 3.2](#) is now available with great new features!
- New [simplified assembly process](#) is now available.

[Tweet](#) [Follow @yanavery](#)





### Features

What features are supported by MyDoorOpener.



### Assembly

How to build and assemble the hardware.



### Config

How to configure the software.

# MyDoorOpener – kody źródłowe

GitHub, Inc. (US) | <https://github.com/yanavery/MyDoorOpener-Arduino/blob/master/MyDoorOpener/MyDoorOpenerServer.cpp> ... Szukaj

yanavery / MyDoorOpener-Arduino

Watch 5 Star 13 Fork 7

Code Issues 1 Pull requests 0 Projects 0 Insights

Branch: master ▼ MyDoorOpener-Arduino / MyDoorOpener / MyDoorOpenerServer.cpp Find file Copy path

yanavery v2.4 94d16ba on 9 Jan 2015

1 contributor

414 lines (317 sloc) | 10 KB Raw Blame History

```
1  //-----
2  // Copyright (C) 2009 - 2014, MyDoorOpener.com
3  //-----
4
5  #include <Ethernet.h>
6  #include <SPI.h>
7  #include "MyDoorOpenerServer.h"
8  #include <aes256.h>
9
10  //#define MYDOOROPENER_SERIAL_DEBUGGING 1
11
```



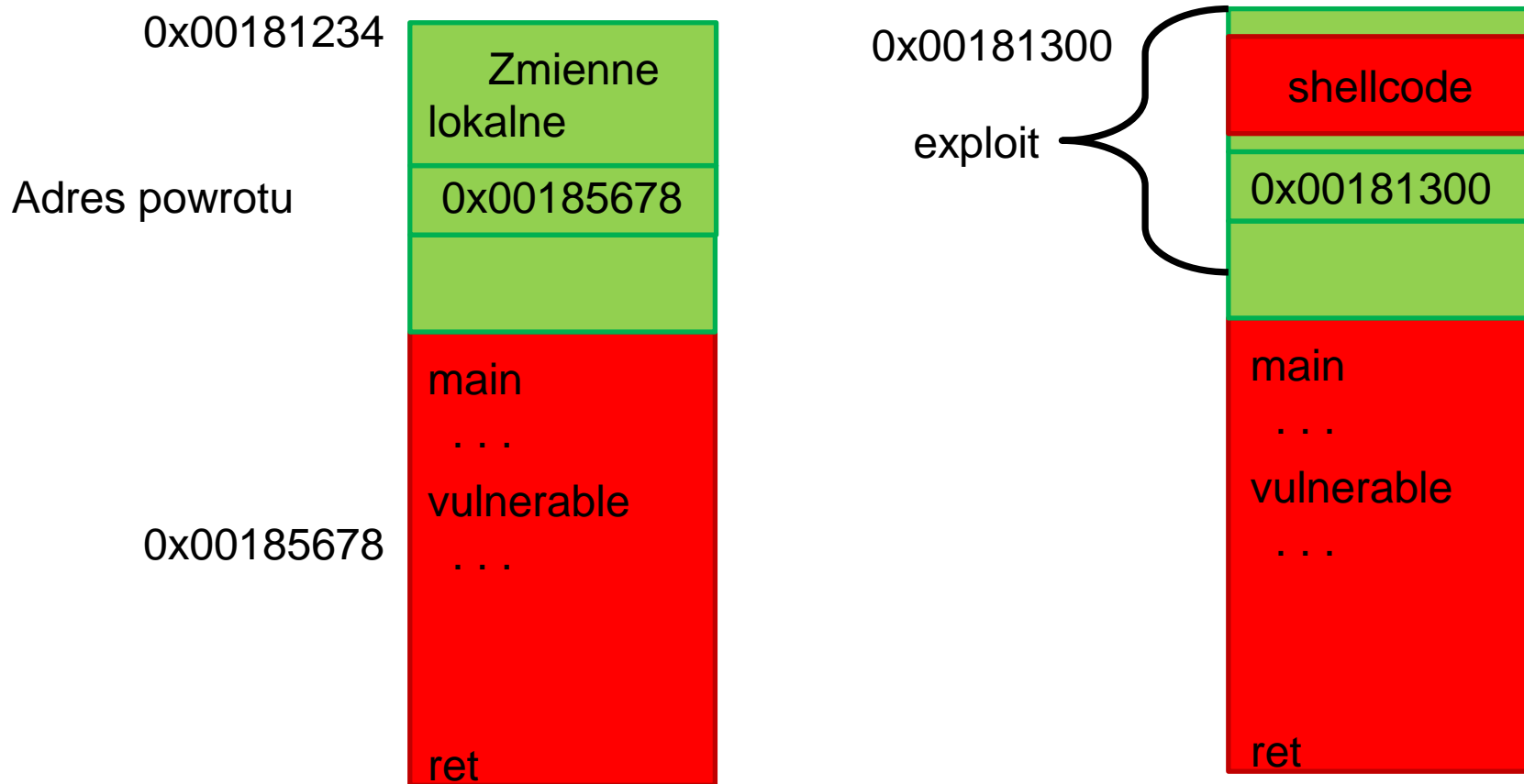
# MyDoorOpener – buffer overflow

```
char request[200];
memset(&request, 0, sizeof(request));

// read client request
readRequest(request);

170 //-----
171 void MyDoorOpenerServer::readRequest(char* url)
172 {
173     while (client.connected())
174     {
175         if (client.available())
176         {
177             char c = client.read();
178
179             if (c == '\n')
180                 break;
181
182             url[strlen(url)] = c;
183         }
184     }
185 }
```

# Stos a wysłanie exploita



# MyDoorOpener

Znaleziony kod został uruchomiony w naszym laboratorium na Arduino wraz z kartą rozszerzeń - Ethernet Shield opartym na układzie W5100

Bezpośrednie uruchomienie kodu na stosie nie jest możliwe na Arduino, ponieważ wykorzystany w nim mikrokontroler Atmega32u4 posiada architekturę Harvard\*

Rozwiązaniem jest wykorzystanie techniki Return Oriented Programing (ROP)

\* Naprawdę zmodyfikowany Harvard, o czym później . . .

# Pokaz ataku

Ewentualnie dalej kilka obrazków jak atak przebiegał w laboratorium ... jeśli coś pójdzie nie tak

# Normalne zachowanie MyDoorOpener

**client request handling begin.**

```
'.quest: 'GET  
  /?relayPin=8&password=1fe063b358bfa8f09cf62742e009064f  
  HTTP/1.1
```

```
submitPassword: '1fe063b358bfa8f09cf62742e009064f'.
```

```
relayPin: '8'.
```

```
submitPassword: '1fe063b358bfa8f09cf62742e009064f'.
```

```
passwordAsChar: 'xxx'.
```

```
Password: 'OK'.
```

```
*** relay triggered ***
```

```
token before: 'Cyber159525'.
```

```
token after: 'Cyber160630'.
```

```
HTTP/1.1 200 OK
```

```
Content-Type: text/xml
```

```
Connection: close
```

```
<?xml version="1.0"?>  
<dooropener>  
<status statusPin="3">*** isOpen - status value  
for pin: '3' is '1' returning: 'Closed' ***  
Closed</status>  
<status statusPin="4">*** isOpen - status value  
for pin: '4' is '1' returning: 'Closed' ***  
Closed</status>  
<challengeToken>Cyber160630</challengeToken>  
</dooropener>
```

**client request handling end.**

# MyDoorOpener podczas ataku

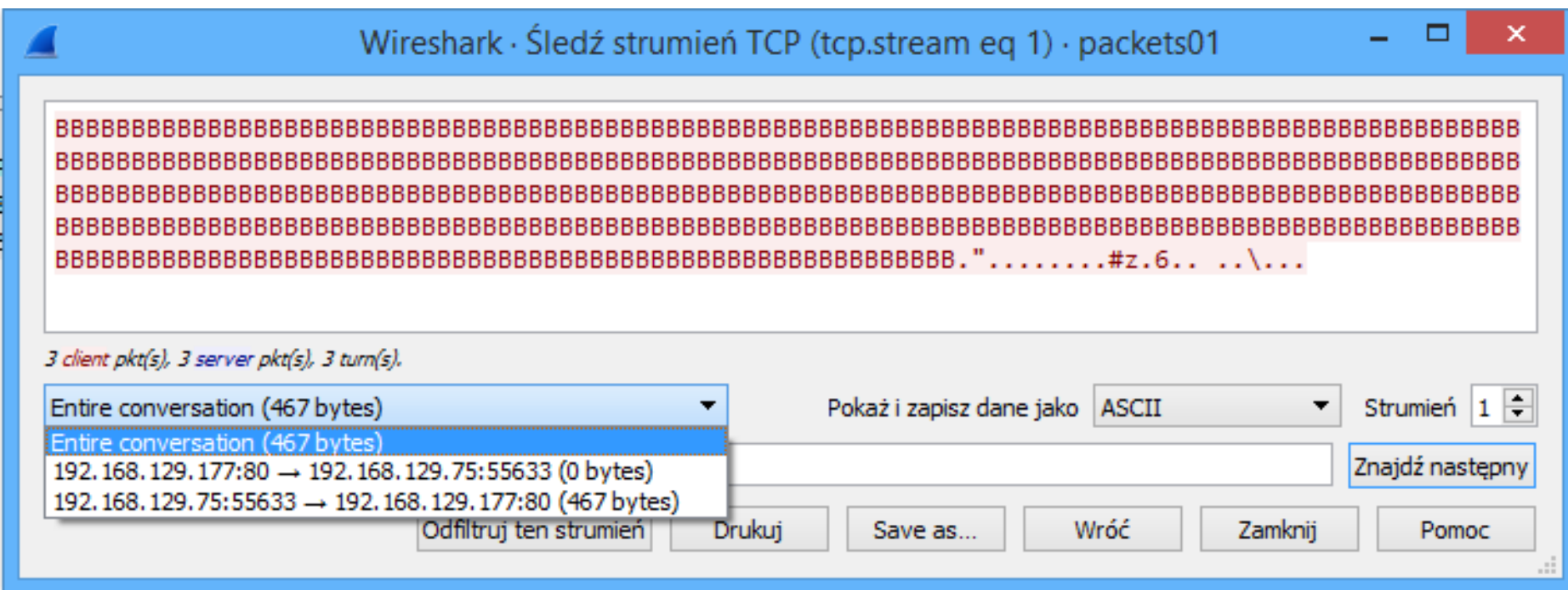
**client request handling begin.**

request:

```
'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'
```

**client request handling end.**

# MyDoorOpener podczas ataku



# Podsumowanie

Udało się nam zaatakować „prawie rzeczywiste” urządzenie ;) – poprzednie prace o podobnej tematyce zakładały, że jest błąd lub działały na własnym kodzie

Prace pokazują, to że nawet na kilkudziesięciu KB udaje się wybrać odpowiednie Gadgets

Po ataku wracamy do dalszej obsługi programu – jedyny efekt uboczny to obniżenie (w kategorii adresów) stosu o 2 bajty



# Podsumowanie

Wyniki zostały opisane w artykule pod tytułem  
**„Compromising an IoT device based on  
Harvard-architecture microcontroller”**  
zgłoszonym na konferencję w Wildze

Kolejne już przygotowane i zweryfikowane ataki:

- przekonfigurowanie trybu portu i potem ustawienie odpowiedniego stanu
- zapisanie dowolnego bajtu pod dowolny adres

# Podsumowanie – dalsze plany

Urządzenie ma możliwość wysyłania powiadomień – wykorzystanie tej funkcjonalności do ataku DoS ( i możliwe, że skanowania?)

Wykorzystanie faktu, że Atmega32u4 posiada w rzeczywistości architekturę zmodyfikowany Harvard i możliwe jest zdalne przeprogramowanie urządzenia – i tutaj możliwości są już prawie nieograniczone ... tj. ograniczone możliwościami tego mikrokontrolera

Automatyzacja wyszukiwania Gadgetów

# Podsumowanie

Czy mają Państwo pytania?