Runtime Security Lab

Michael Schwarz
September 16, 2019
Security Week Graz 2019
Large IoT Incidents

- September 21, 2016
  > 600 Gbps on Brian Krebs (security researcher) website (Mirai botnet)

- September 30, 2016
  Mirai source code published

- October 21, 2016
  ~1 Tbps on DNS provider Dyn

- November 26, 2016
  > 900 000 routers of Deutsche Telekom attacked and offline

- February, 2018
  > 1.35 Tbps attack on GitHub
Top 10 IoT Bugs

HELPING SECURE THE INTERNET OF THINGS WITH THE

OWASP

INTERNET OF THINGS

VULNERABILITY CATEGORIES

Michael Schwarz — Security Week Graz 2019
Top 10 IoT Bugs

1. Insecure Web Interface

Default usernames and passwords
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services

Unnecessary ports open
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism

Updates are not signed
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Insecure or Outdated Components

Software with security vulnerabilities
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Insecure or Outdated Components
6. Insufficient Privacy Protection

Collected information not properly protected
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage

SSL/TLS not available
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management

No device monitoring
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management
9. Insecure Default Settings
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management
9. Insecure Default Settings
10. Lack of Physical Hardening

Unnecessary external ports like USB
The 90s called...
The 90s called...

...they want their bugs back!
HACK

ALL THE THINGS!
There are 19 challenges
- There are 19 challenges
- Different difficulties (the more points, the harder)
• There are 19 challenges
• Different difficulties (the more points, the harder)
• 4 different categories
- There are 19 challenges
- Different difficulties (the more points, the harder)
- 4 different categories
- Play on your own or as team
• Capture-the-flag (CTF) style
• Capture-the-flag (CTF) style
• Every challenge has a hidden flag
• Capture-the-flag (CTF) style
• Every challenge has a hidden flag
• Flags are usually in a text file flag.txt on the device
Infrastructure

- Capture-the-flag (CTF) style
- Every challenge has a hidden flag
- Flags are usually in a text file flag.txt on the device
- A flag looks like CTF{A_S4MPL3_FL4G!}
• Capture-the-flag (CTF) style
• Every challenge has a hidden flag
• Flags are usually in a text file flag.txt on the device
• A flag looks like CTF{A_S4MPL3_FL4G!}
• Goal is to get the flag and submit it to the CTF system
CTF runs until Friday, 11:59am
- CTF runs until Friday, 11:59am
- Last-minute questions from 11:00am to 11:59am
Timeline

- CTF runs until **Friday, 11:59am**
- Last-minute questions from 11:00am to 11:59am
- Best player/team gets a **price**
How to Start

- Use your own computer or our provided Linux VM (on USB or from https://ctf.attacking.systems/rtfm)

Create or join a team in the CTF system: https://ctf.attacking.systems

Choose a hacklet, read the description, and download it

Solve the hacklet, submit the flag in the CTF system
How to Start

- Use your own computer or our provided Linux VM (on USB or from https://ctf.attacking.systems/rtfm)
- Create or join a team in the CTF system: https://ctf.attacking.systems
How to Start

- Use your own computer or our provided Linux VM (on USB or from https://ctf.attacking.systems/rtfm)
- Create or join a team in the CTF system: https://ctf.attacking.systems
- Choose a hacklet, read the description, and download it
How to Start

- Use your own computer or our provided Linux VM (on USB or from https://ctf.attacking.systems/rtfm)
- Create or join a team in the CTF system: https://ctf.attacking.systems
- Choose a hacklet, read the description, and download it
- Solve the hacklet, submit the flag in the CTF system
How to Connect

- Some hacklets are accessible over the network
How to Connect

- Some hacklets are accessible over the network
- These hacklets have a text interface on a specific port

Example: netcat hacklets2.attacking.systems 8000
How to Connect

- Some hacklets are accessible over the network
- These hacklets have a text interface on a specific port
- You can connect using any telnet-like program:
  - Windows: PuTTY
  - Apple: Terminal, netcat, telnet
  - Linux: netcat, telnet

For example on Linux/Mac in the shell:
```
netcat hacklets2.attacking.systems 8000
```
How to Connect

- Some hacklets are accessible over the network
- These hacklets have a text interface on a specific port
- You can connect using any telnet-like program:
  - PuTTY
  - Terminal, netcat, telnet
  - netcat, telnet
- For example on Linux/Mac in the shell:
  ```
  netcat hacklets2.attacking.systems 8000
  ```
There are 4 categories: **binary**, **crypto**, **formats**, **misc**
There are 4 categories: binary (ภา), crypto ( webdriver ), formats (تقنية）， misc (_misc)
There are 4 categories: binary (◎), crypto (🔍), formats (📖), misc ( nodo)

◎ Vulnerable/insecure binaries which you have to exploit
🔍 Bad/Misused cryptography you have to break
The Categories

There are 4 categories: **binary**, **crypto**, **formats**, **misc**

- Vulnerable/insecure binaries which you have to exploit
- Bad/Misused cryptography you have to break
- Understanding custom formats
There are 4 categories: binary (℞), crypto (🔍), formats (📑), misc (攸)

℞ Vulnerable/insecure binaries which you have to exploit
🔍 Bad/Misused cryptography you have to break
📑 Understanding custom formats
攸 Random and fun hacklets which do not fit into any category
  (often no programming required)
REGISTER FOR CTF

GET ALL THE HACKLETS

HOW TO START

HOW TO START
Download the hacklet
How to Start

- Download the hacklet
- Identify the type of file
  - 🌟 Executable? For which platform?
  - 📁 Data? Which program can open it?
  - 🔄 Unknown?
How to Start

- Download the hacklet
- Identify the type of file
  - 🌟 Executable? For which platform?
  - 📁 Data? Which program can open it?
  - 🕵️‍♂️ Unknown?
- Useful Linux tool: `file` – determines the file type
• Maybe file is some archive...
• Maybe file is some archive...
• ...or contains **multiple files**
• Maybe file is some archive...
• ...or contains **multiple files**
• Binwalk Firmware Analysis Tool
  🌐 [https://github.com/ReFirmLabs/binwalk](https://github.com/ReFirmLabs/binwalk)
• Maybe file is some archive...
• ...or contains **multiple files**
• Binwalk Firmware Analysis Tool
  🌐 https://github.com/ReFirmLabs/binwalk
• Can also **extract** files
• Run `strings` on the file to extract all texts
• Run **strings** on the file to extract all texts

• For binaries: see all functions/variables (i.e., symbols)
  - x86: `objdump -x <hacklet>`
  - ARM: `arm-linux-gnueabihf-objdump -x <hacklet>`
• Run `strings` on the file to extract all texts
• For binaries: see all functions/variables (i.e., symbols)
  • x86: `objdump -x <hacklet>`
  • ARM: `arm-linux-gnueabihf-objdump -x <hacklet>`
• Watch out for function names containing flag
Binaries

- Try to run the binary
  - x86: no requirements
  - ARM: requires
    
    libc6-dev-armhf-cross qemu-system-arm qemu-user

- More details:
  - https://ctf.attacking.systems/rtfm

- Use a network monitor (Wireshark) to detect connections
• Try to run the binary
  • x86: no requirements
  • ARM: requires
    libc6-dev-armhf-cross qemu-system-arm qemu-user

• Then simply execute
  qemu-arm -L /usr/arm-linux-gnueabihf ./hacklet
  or for ARMv8
  qemu-aarch64 -L /usr/aarch64-linux-gnu ./hacklet
Binaries

- Try to run the binary
  - x86: no requirements
  - ARM: requires
    ```
    libc6-dev-armhf-cross qemu-system-arm qemu-user
    ```

- Then simply execute
  ```
  qemu-arm -L /usr/arm-linux-gnueabihf ./hacklet
  ```
  or for ARMv8
  ```
  qemu-aarch64 -L /usr/aarch64-linux-gnu ./hacklet
  ```

- More details: https://ctf.attacking.systems/rtfm
Try to run the binary
- x86: no requirements
- ARM: requires
  libc6-dev-armhf-cross qemu-system-arm qemu-user

Then simply execute
  qemu-arm -L /usr/arm-linux-gnueabihf ./hacklet
  or for ARMv8
  qemu-aarch64 -L /usr/aarch64-linux-gnu ./hacklet

More details: https://ctf.attacking.systems/rtfm

Use a network monitor (Wireshark) to detect connections
• Command-line disassembler
  • x86: `objdump -d <hacklet>`
  • ARM: `arm-linux-gnueabi-objdump -d <hacklet>`
  • All platforms: `radare2`
• Command-line disassembler
  - x86: `objdump -d <hacklet>`
  - ARM: `arm-linux-gnueabi-objdump -d <hacklet>`
  - All platforms: radare2
• Watch out for dangerous functions (e.g. `strcpy`, `gets`)

Michael Schwarz — Security Week Graz 2019
Reverse Engineering

- Command-line disassembler
  - x86: `objdump -d <hacklet>`
  - ARM: `arm-linux-gnueabi-objdump -d <hacklet>`
  - All platforms: radare2
- Watch out for dangerous functions (e.g. `strcpy`, `gets`)
- GUI disassembler: cutter
  🌐 [https://github.com/radareorg/cutter](https://github.com/radareorg/cutter)
• Decompiler generates (pseudo) code from binary
Decompiler generates (pseudo) code from binary
Easier to understand what a binary does
Decompiler generates (pseudo) code from binary
Easier to understand what a binary does
GUI decompiler: Ghidra

https://ghidra-sre.org/
Decompiler generates (pseudo) code from binary
Easier to understand what a binary does
GUI decompiler: Ghidra
Open source, supports many architectures
• It helps to explain what you see
• It helps to **explain** what you see
• Talking about the problem can be the first step
• It helps to explain what you see
• Talking about the problem can be the first step
• Usually we talk to humans
Rubberduck Debugging

- It helps to explain what you see
- Talking about the problem can be the first step
- Usually we talk to humans
- If none available/interested: use a rubber duck!
Let’s start with the challenges!
What next?

- Let's start with the challenges!
- https://ctf.attacking.systems
What next?

- Let’s start with the challenges!
- https://ctf.attacking.systems
- If you are unsure, there is a walkthrough of one hacklet: https://ctf.attacking.systems/rtfm
What next?

- Let’s start with the challenges!
- https://ctf.attacking.systems
- If you are unsure, there is a walkthrough of one hacklet: https://ctf.attacking.systems/rtfm
A Challenge a Day Keeps the Boredom Away
Questions?