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
BUGS

BUGS EVERYWHERE
Top 10 IoT Bugs

Helping secure the Internet of Things with the OWASP Top 10 Vulnerability Categories.
Top 10 IoT Bugs

1. Insecure Web Interface

Default usernames and passwords
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication

Weak passwords
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption

SSL/TLS not available
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns

Collected information not properly protected
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns
6. Insecure Cloud Interface

Interfaces with security vulnerabilities
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns
6. Insecure Cloud Interface
7. Insecure Mobile Interface
8. Insufficient Security Configurability
9. Insecure Software/Firmware
10. Poor Physical Security

No account lockout mechanisms
1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns
6. Insecure Cloud Interface
7. Insecure Mobile Interface
8. Insufficient Security Configurability

Encryption is not available
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns
6. Insecure Cloud Interface
7. Insecure Mobile Interface
8. Insufficient Security Configurability
9. Insecure Software/Firmware
10. Poor Physical Security

Updates are not signed
Top 10 IoT Bugs

1. Insecure Web Interface
2. Insufficient Authentication
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns
6. Insecure Cloud Interface
7. Insecure Mobile Interface
8. Insufficient Security Configurability
9. Insecure Software/Firmware
10. Poor Physical Security

Unnecessary external ports like USB

Michael Schwarz — Security Week Graz 2018
The 90s called...
The 90s called...

...they want their bugs back!
HACK

ALL THE THINGS!
• There are 15 challenges
• There are 15 challenges
• Different difficulties (the more points, the harder)
There are 15 challenges
- Different difficulties (the more points, the harder)
- 4 different categories
There are 15 challenges
Different difficulties (the more points, the harder)
4 different categories
Play on your own or as team
<table>
<thead>
<tr>
<th>Category</th>
<th>Challenge</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pwn</strong></td>
<td>Warm Up</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Secure Webserver</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Debug Shell I</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Debug Shell II</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>dJIT Integrator</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>dJIT Plot</td>
<td>150</td>
</tr>
<tr>
<td><strong>misc</strong></td>
<td>RTFM</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Weird Architecture</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Secure Router</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Let's Play a Game</td>
<td>50</td>
</tr>
<tr>
<td><strong>forensics</strong></td>
<td>Printer Update</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Random Blob</td>
<td>50</td>
</tr>
<tr>
<td><strong>crypto</strong></td>
<td>Flight Radar</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lightweight Crypto</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Encrypted Blob</td>
<td>50</td>
</tr>
</tbody>
</table>
• 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
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 \{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 {A_S4MPL3_FL4G!}
• Goal is to get the flag and submit it to the CTF system
• CTF runs until Friday, 3:00pm
Timeline

- CTF runs until Friday, 3:00pm
- Last-minute questions from 2:00pm to 3:00pm
• CTF runs until Friday, 3:00pm
• Last-minute questions from 2:00pm to 3:00pm
• 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/res)
How to Start

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

- 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/resources)
- 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/res)
- 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 by connecting to the hacklet
• Hacklets are accessible over the network
• Hacklets are accessible over the network
• Every hacklet has a text interface on a specific port
How to Connect

- Hacklets are accessible over the network
- Every hacklet has a text interface on a specific port
- You can connect using any telnet-like program:
  - PuTTY
  - Terminal, netcat, telnet
  - netcat, telnet
How to Connect

- Hacklets are accessible over the network
- Every hacklet has a text interface on a specific port
- You can connect using any telnet-like program:
  - Windows: PuTTY
  - macOS: Terminal, netcat, telnet
  - Linux: netcat, telnet
- For example on Linux/Mac in the shell:
  
  netcat hacklets2.attacking.systems 8000
There are 4 categories: pwn (💀), forensics (🔍), crypto (🔑), misc (💡)
There are 4 categories: pwn (☠️), forensics (🔧), crypto (🔑), misc (杵)

☠️ Vulnerable binaries which you have to exploit
There are 4 categories: pwn (💀), forensics (🔍), crypto (キー), misc (❓)

💀 Vulnerable binaries which you have to exploit

🔍 Basically finding/reconstructing hidden/deleted stuff
The Categories

There are 4 categories: pwn (_skull_), forensics (_pi-day_), crypto (_key_), misc (_head_)

💀 Vulnerable binaries which you have to exploit
🔍 Basically finding/reconstructing hidden/deleted stuff
🔍 (Bad) Cryptography you have to break
There are 4 categories: pwn (💀), forensics (🔍), crypto (🔍), misc (🧠)

💀 Vulnerable binaries which you have to exploit
🔍 Basically finding/reconstructing hidden/deleted stuff
🔍 (Bad) Cryptography you have to break
🧠 Random and fun hacklets which do not fit into any category
    (often no programming required)
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
• 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-gnueabi-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
```

- 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/res

- Use a port scanner to check for alternative interface (SSH is not exploitable!)
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`
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/res
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/res

- Use a port scanner to check for alternative interface
  (SSH is not exploitable!)
Reverse Engineering

- Command-line disassembler
  - x86: `objdump -d <hacklet>`
  - ARM: `arm-linux-gnueabihf-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)
- 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`)
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)
Rubberduck Debugging

- 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
• 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*!
What next?

- 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](https://ctf.attacking.systems)
- If you are unsure, there is a **walkthrough** of one hacklet: [https://ctf.attacking.systems/res](https://ctf.attacking.systems/res)
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/res
A Challenge a Day Keeps the Boredom Away
Questions?