Cotopaxi

IoT protocols testing toolkit
Jakub Botwicz
• Principal Security Engineer at Samsung R&D Institute in Warsaw, Poland
• Leads a team of security researchers / pentesters
• PhD and MSc at Warsaw University of Technology
• 16+ years experience - previously worked as:
  – Developer/architect for vendor of encryption devices
  – Security advisor at credit card payment company
  – Security consultant and manager at Big4 company
• Over 35 CVEs reported in recent 3 years
• Big enthusiast of rock climbing and active volcanoes
Cotopaxi – contributors

• Anonymous Cat
• liyansong2018
• Mariusz Księżak
• Michał Radwański
• Wiktor Gerstenstein

Source: https://twitter.com/mmatp/status/1151851535439159299/photo/1
• IoT introduced **new protocols**: CoAP, DTLS, MQTT and refurbished old protocols: UPnP, SSDP

• Lack of **security testing tools** for IoT protocols (except for MQTT)

• Low level of **security measures** of IoT components and devices

• Our team performed assessment of multiple IoT components – results:  
  - ideas for tools  
  - corpus of malformed messages  
  - 20+ new vulnerabilities
IoT security tool – landscape (end of 2019)

Wireless stacks
- ZigBee
- Bluetooth
  - BLESuite
  - GATTacker
  - ZigDiggity
  - Other BT/BLE tools

Network tools
- Transport layer
  - DTLS
  - QUIC
- Application layer
  - AMQP
  - CoAP
  - MQTT
  - mDNS
  - RTSP
  - SSDP

Exploit packs
- ExploT
- HomePWN
- IoTSecFuzz

Hardware hacking tools
- UART
- JTAG

Firmware tools
- FACT
- Bus Pirate
- JTAGulator
- Firmware Auditor
- Other RE tools

(*) only for public devices
# IoT protocols supported by free IoT tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>AMQP</th>
<th>CoAP</th>
<th>DTLS</th>
<th>HTCPCP</th>
<th>mDNS</th>
<th>MQTT</th>
<th>QUIC</th>
<th>RTSP</th>
<th>SSDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_ping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>server_fingerprint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>credential_cracker</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resource_listing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>protocol_fuzzer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>client_proto_fuzzer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vulnerability_tester</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>client_vuln_tester</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amplifier_detector</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>active_scanner</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*New tool (2018+)*

*Stable/maintained*  
*Abandoned tool*
**nmap scan on CoAP / DTLS servers**

```
sudo nmap 1xx.1xx.1xx.1xx -sU -p 10001-10005,20000-20010 -A
```

Starting Nmap 7.70 (https://nmap.org) at 2019-99-9 99:99 CET
Nmap scan report for 1xx.1xx.1xx.1xx
Host is up (0.00059s latency).

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>open</td>
<td>filtered</td>
<td>scp-config</td>
</tr>
<tr>
<td>10002</td>
<td>open</td>
<td>documentum?</td>
<td></td>
</tr>
<tr>
<td>10003</td>
<td>open</td>
<td>documentum_s?</td>
<td></td>
</tr>
<tr>
<td>10004</td>
<td>open</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>10005</td>
<td>open</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>open</td>
<td>filtered</td>
<td>dnp</td>
</tr>
<tr>
<td>20001</td>
<td>open</td>
<td>filtered</td>
<td>microsan</td>
</tr>
<tr>
<td>20002</td>
<td>open</td>
<td>commtact-http</td>
<td></td>
</tr>
<tr>
<td>20003</td>
<td>open</td>
<td>commtact-https</td>
<td></td>
</tr>
<tr>
<td>20005</td>
<td>open</td>
<td>filtered</td>
<td>openwebnet</td>
</tr>
<tr>
<td>20006</td>
<td>open</td>
<td>filtered</td>
<td>unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>62 108.72</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10002</td>
<td>12</td>
</tr>
<tr>
<td>63 111.35</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10002</td>
<td>12</td>
</tr>
<tr>
<td>64 118.01</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10003</td>
<td>12</td>
</tr>
<tr>
<td>65 120.32</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10003</td>
<td>12</td>
</tr>
<tr>
<td>70 132.58</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10001</td>
<td>12</td>
</tr>
<tr>
<td>71 135.20</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10001</td>
<td>12</td>
</tr>
<tr>
<td>80 144.32</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10001</td>
<td>12</td>
</tr>
<tr>
<td>81 147.26</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10001</td>
<td>12</td>
</tr>
<tr>
<td>86 157.34</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>1001</td>
<td>12</td>
</tr>
<tr>
<td>87 162.27</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10001</td>
<td>12</td>
</tr>
<tr>
<td>88 166.10</td>
<td>106.1 106.1</td>
<td>UDP</td>
<td>54</td>
<td>10101</td>
<td>12</td>
</tr>
</tbody>
</table>

User Datagram Protocol, Src Port: 48443, Dst Port: 10002

Data (12 bytes)

Data: 4001736a7227124474657374
[Length: 12]

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>ASCII</th>
</tr>
</thead>
</table>
| 00 08 e3 ff fd 90 50 65 f3 2a 28 5b 08 00 45 00 | ······Pe *(···E.
| 00 28 3b 89 40 00 40 11 e9 a2 6a 78 b8 11 6a 78 | (;·@···jx·jx
| 00 20 88 97 bd 3b 27 12 00 14 15 bf 40 01 73 6a 72 27 | ...'····@·sjr'
| 00 30 12 44 74 65 73 74 | ·Dtest
Wireshark view on CoAP traffic – “Decode As” CoAP

User Datagram Protocol, Src Port: 48443, Dst Port: 10002

Constrained Application Protocol, Confiable, GET, MID:29546

01.. .... = Version: 1
..00 .... = Type: Confiable (0)
.... 0000 = Token Length: 0
Code: GET (1)
Message ID: 29546

Opt Name: #1: Uri-Port: 10002

Opt Desc: Type 7, Critical, Unsafe
0111 .... = Opt Delta: 7
.... 0010 = Opt Length: 2
Uri-Port: 10002

Opt Name: #2: Uri-Path: test

Opt Desc: Type 11, Critical, Unsafe
0100 .... = Opt Delta: 4
.... 0100 = Opt Length: 4
Uri-Path: test

[Uri-Path: :10002/test]

00000 00 00 e3 ff fd 90 50 65 f3 2a 28 5b 08 00 45 00          Pe /*(E ...
00010 00 28 39 fd 40 00 40 11 eb 2e 6a 78 b8 11 6a 78 (9  @  .)x .x ...
00020 88 97 bd 3b 27 12 00 14 15 bf 40 01 73 6a 72 27 ....;.....@s'jr'
00030 12 44 74 65 73 74 Dtest
• IoT introduced **new protocols**: CoAP, DTLS, MQTT and refurbished old protocols: UPnP, SSDP

• Lack of **security testing tools** for IoT protocols (except for MQTT)

• Low level of **security measures** of IoT components and devices

• Our team performed assessment of multiple IoT components – results:
  - ideas for tools
  - corpus of malformed messages
  - 20+ new vulnerabilities
Cotopaxi – the toolkit

- Set of tools for security testing of Internet of Things devices using network IoT protocols
- License: GPL-2.0
- Repository: https://github.com/samsung/cotopaxi
- Releases:
  - 1st release in March 2019 (before Black Hat Asia)
  - 4th release in August 2020 (before Black Hat USA and DEFCON)
Cotopaxi – the name

- Active stratovolcano in Ecuador
- Elevation: 19,347 ft / 5,897 m
Cotopaxi – usage scenarios

• For pentesters to:
  • analyze environments using IoT, Smart-\{Home, Factory, City\}
  • find active endpoints using IoT protocols
  • classify software components or IoT devices
  • identify network traffic reflectors (DDoS)

• For security researchers to:
  • perform „black box” testing of IoT devices
  • identify known security vulnerabilities
  • identify OEM devices (by classification)

• For developers or vendors of IoT devices to:
  • fuzz components or interfaces
  • test traffic amplification (DDoS)
Reconnaissance phase:

- **Service ping** - checking availability of network services
- **Security scanner** - verifying security settings (e.g. supported ciphersuites, certificates)
- **Software fingerprinting** - recognizing the software used by remote network server
- **Resource listing „dirbusting”** - discovering resources identified by given URLs
- **Device identification** - passive analysis of traffic and device classification using Machine Learning

Pre-exploitation phase:

- **Amplification sniffing** - detecting network traffic amplification
- **Protocol fuzzing** - fuzzing implementation of protocol
  - From 2nd release for protocol clients and servers
- **Vulnerability testing** - identifying known vulnerabilities
  - From 2nd release for protocol clients and servers

Source: https://www.amazon.com/Wenger-16999-Swiss-Knife-Giant/dp/B001DZTJRQ
Cotopaxi – supported IoT protocols

- **Supported in 1st release:**
  - Constrained Application Protocol (CoAP)
  - Datagram Transport Layer Security (DTLS)
  - Multicast DNS (mDNS) / DNS-System Discovery (DNS-SD)
  - Message Queuing Telemetry Transport (MQTT)

- **Added in 2nd and 3rd releases:**
  - Hyper Text Coffee Pot Control Protocol (HTCPCP)
  - Simple Service Discovery Protocol (SSDP)
  - Real Time Streaming Protocol (RTSP)

- **Added in 4th release:**
  - Advanced Message Queuing Protocol (AMQP)
  - MQTT-Sensor Networks (MQTT-SN)
  - Quick UDP Internet Connections (QUIC)

- **Planned for next releases:**
  - Distributed Network Protocol (DNP3)
  - KNX (building automation protocol)
IoT protocols – Quick UDP Internet Connections (QUIC)

• Created by Google and widely used in Google Apps like Docs, Maps and IoT devices
• Transport layer for HTTP/3
• General-purpose transport layer protocol with:
  - built-in communication security
  - multiple streams in one connection
  - low latency on setup
• UDP-based

Source: Kate Pearce „HTTP/2 & QUIC Teaching Good Protocols To Do Bad Things“
Message Queuing Telemetry Transport – Sensor Networks

- **MQTT**
  - most popular IoT messaging protocol
  - TCP-based

- **MQTT-SN (Sensor Networks)**
  - UDP-based clone of MQTT
  - small changes in packet formats
  - all ideas are the same
  - not as popular as MQTT

Source: Trend Micro „The Fragility of Industrial IoT’s Data Backbone”
• Identifies active service endpoints (IPv4/IPv6:port)

• Not uses ICMP echo!

• For each protocol there is a set of messages that triggers responses in all tested servers
  e.g. DTLS – Client Hello in all DTLS versions

• Better than using standard tools: nmap and wireshark do not recognize IoT traffic on non-standard ports
Detection of software and version used by server

Equivalent to `nmap -sV` (Service and Application Version Detection)

Uses machine learning classifier

In this version works only for CoAP, DTLS

Cotopaxi feature – device identification

• Identification of IoT devices using captured traffic (PCAP format) using machine learning classifier

• In this version supports 60+ devices: webcameras, SmartHome devices, SmartTVs

• Corpus of traffic for IoT devices provided by authors of papers:

  Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi

  A. Sivanathan, H. Habibi Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath and V. Sivaraman,
  “Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics”

```
[.] Started classification for IP: 13.37.0.17
[!] Found 1000 packets to or from this IP
[*] Device was classified as: Xiaomi Mi Cam 2
[*] Results of device identification:
  Xiaomi Mi Cam 2: 59.09%
  Amcrest Camera: 36.87%
  Wansview Camera: 1.92%
  Yi Camera: 1.62%
  Bosiwo Camera: 0.20%
  Luohe Cam: 0.20%
  Lefun Cam: 0.10%
[.] Classification time: 3.19 sec
```
Cotopaxi feature – resource listing (dirbusting)

• Equivalent to:
  • *DirBuster/dirb* for CoAP/mDNS/SSDP
    nmap script *coap-resources*

• Uses a list of:
  • URIs (for CoAP)
  • services (for mDNS and SSDP)
  • media files (for RTSP)

• Cotopaxi includes sample lists of resource

• User can provide own list of URIs or services

Source: https://medium.com/coinbundle/blockchain-internet-of-things-iot-be58703617c9
Cotopaxi feature – protocol fuzzer (black-box)

- Uses corpus of malformed protocol messages (payloads) prepared with afl (American Fuzzy Loop)
- Checks „service ping” before and after sending payload
- Allows to use own corpus of payloads and integrate with mutating fuzzer
- In verbose mode displays payload and response packet
- Calculates RTT for payloads with responses and displays Top 10% - potentially interesting because of longest processing on server

<table>
<thead>
<tr>
<th>Protocol</th>
<th>AMQP</th>
<th>CoAP</th>
<th>DTLS</th>
<th>mDNS/DNS-SD</th>
<th>MQTT</th>
<th>HTCP</th>
<th>SSDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of corpus</td>
<td>147</td>
<td>557</td>
<td>395</td>
<td>77</td>
<td>285</td>
<td>469</td>
<td>271</td>
</tr>
</tbody>
</table>
Types of vulnerabilities:

- **information disclosure** – unauthorized access to internal information
- **crash (DoS)** – leads to crash of server (detected by service ping)
- **traffic amplification (for DDoS)** – responses larger than request
- **memory leak** – server wastes memory after processing payload (requires manual confirmation)
- **remote code execution** (currently only detected as crashes)

In this release 10 vulnerabilities!

* some vulnerabilities are currently in responsible disclosure process
Example bugs found by our team and identified by Cotopaxi:

- **CVE-2019-9747** in tinysvcmdn (hang)
  mDNS server goes into infinite loop after receiving DNS query with recursive referenced names

- **CVE-2019-9004** in Wakaama (memory leak)
  CoAP server leaks (wastes) 24 bytes per each processed crafted packet

- **CVE-2019-9750** in IoTivity (traffic amplification)
  CoAP server responds with 6 error messages

- **CVE-2019-18840** in WolfSSL (crash)
  TLS and DTLS servers and clients can be crashed using malformed x509 certificate
• Sniffs for all packets incoming to and outgoing from specified target

• Calculates amplification factor (size_out/size_in – 1)

• Tracks req/resp with highest amplification factor and display record on exit

• Should be placed on router or use network tap to see all traffic to/from target

Source: http://m.quickmeme.com/Rat-Sniffer/
DDoS attack via IP Spoofing and Traffic Amplification

- Identified amplifiers-reflectors:
  - **CoAP**
    - every request with large response
      - e.g. CoAPthon example /big
    - IoTivity 4.04 6x repeated response
      - IOTIVITY_000 issue in tester
  - **DTLS**
    - every DTLS server without Hello Verify Request
      - e.g. Botan_000 issue in tester

Source: Austin Brooks „NTP DDoS Vulnerability“
Cotopaxi – next steps

- Download from Samsung GitHub
  https://github.com/samsung/cotopaxi

- **Hack the planet!**
  ...but only with the written consent of the owner :-)

- Read the friendly manual in Readme.md

- Report any issues or requests for new features on GitHub

- Pull request with new features

Source: https://while2.ghost.io/rtfmc/