# IOT Pentesting Test Cases - 2019

| S.No | Category | Type | Stage 1 - Enum | Stage 2 - Enum | Stage 3 - Enum | Stage 4 - Enum |
|---|---|---|---|---|---|---|
| 1 | Firmware | Non-Encrypted Part -1 | Extract file system using Binwalk (binwalk -e firmware.bin) | Analyze configuration files and hardcoded sensitive values and certificates (manually + automated tool - Firmwalker) | Identify vulnerabilities such as command injection and backdoors in the disassemblers | |
| | | | | | Analyze strings present in the binary | |
| | | | | Disassemble individual binaries using Hopper/ Binary Ninja/IDA Pro | Analyze the function list | |
| | | | | | Look for Xrefs to system() | |
| | | | Emulate individual binaries using qemu (sudo chroot . ./qemu-arch -L optional-lib-path -g gdb-port binary-to-emulate) | | Set breakpoints at functions like strcmp and analyze the context (registers, stack and disassembly) at that point. | |
| | | | | Attach GDB-Multiarch (or IDA Pro) to the emulated binary | Identify overflow based vulnerabilities (pass a large string and see if the program crashes) | Exploit the overflow by forming a ROP chain (use ROPGadget to find useful gadgets) |
| | | Non-Encrypted Part -2 | Modify firmware image using FMK (Firmware-Mod-Kit) | ./extract-firmware.sh to extract the firmware | Build the firmware using build-firmware.sh with the flags -nopad -min | |
| | | | | Upload the modified version of the firmware to the target device | If the DUT accepts the new firmware, it is missing firmware integrity verification | |
| | | | | | If it does not accept, look for code sample in the firmware (or other components) where it's checking for signature verification. | |
| 2 | | Obtaining the firmware | Vendor website and support forums | | | |
| | | | Sniffing the package during OTA update | | | |
| | | | Reversing mobile application | | | |
| | | | Dumping from the device using H/W Exploitation tactics | | | |
| 3 | Hardware | Recon | FCC-ID database : Look up the FCC ID of the DUT on fccid.io | | | |
| | | | Public searching of previous accessible resources of the device | Look for research done on previous versions of the device if that's available to get an idea of the process | | |
| | | Teardown | Identify the kind of enclosure holding the packaging together and use appropriate tools to open it | Look for screws beneath the rubber pads | | |
| | | | | Pry open the device | | |
| | | | | Apply heat (make sure to not damage the device) | | |
| | | | | Cut/Remove the enclosure (as the last option) | | |
| | | | | XRays/Focused Ion Beam Workstation | | |
| | | Exploitation | What does the PCB Reveal | Identify different chips and part numbers | | |
| | | | | | UART - usually in pair of 3 or 4 with one of the | |
| | | | | | pins being GND 2 | |
| | | | | | SPI - presence of a flash chip 2 | |
| | | | | | JTAG - set of 6,12,13,20 pin headers (or could also | |
| | | | | | be scattered across the board) | |
| | | | | | | Research the chip to identify the possible communication protocols |
| | | | | Can you see any possible interfaces? | Unfamiliar protocol | Use a logic sniffer to see what data is being communicated (can use different analyzers and see which one shows something meaningful) |
| | | | | | External interfaces like USB (can plug in a keyboard and brute force with special key combinations) | |
| | | Backdooring | Upload malicious firmware to the device using Flash re-write or using JTAG | | | |
| | | | Add your own new component for persistent hardware backdooring on the device | | | |
| | | | Find out information such as how the firmware is downloaded from the remote endpoint and flashed to the device | | | |
| | | | Protocol and encryption (key could be found here or by dumping the flash contents in h/w) used for communication | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4 | Conventional | Mobile | Reverse the API communication | | | |
| | | | | What kind of data is being stored on the mobile device | | | |
| | | | | Use Frida to perform runtime manipulation and analysis | | | |
| | | | | Automated mobile application security tools | | | |
| | | | | Replay based attacks | | | |
| | | | | Insufficient authentication and authorization checks when communication with the device | | | |
| | | | Web | IDOR (Insecure Direct Object Reference) | | | |
| | | | | Check for permission level bugs (admin, user, superadmin) | | | |
| | | | | Typical bugs - XSS, SQLi, XXE, XSRF etc. | | | |
| | | | Network | Check for running services on the device | | | |
| | | | | Outdated services | | | |
| | | | | Password brute-forcing and cracking | | | |
| | | | | Unfamiliar port open? | | | |
| | | | | Sniff the network communication | | | |
| | | | Thick Clients | RE the Thick Client to find vulnerabilities such as Command Injection and Overflows | | | |
| | | | | Communication to the remote endpoint | | | |
| | | | | Reversing the APIs | | | |
| | | | | Exploiting Trust components of the Thick Client | | | |
| | 5 | Radio | Raw Radio Communication protocol | Identify the frequency using HackRF / RTL-SDR ( Notice the spikes in GQRX when device sends bursts of data) | Capture the data being transmitted at that frequency and process it in GNURadio to obtain meaningful information | | |
| | | | | | Use hackrf_transfer to replay the captured data | | |
| | | | BLE | Identify the BLE devices around you and their addresses (using a BLE dongle) | Identify the BLE devices around you and their addresses (using a BLE dongle) | Clear-text traffic | |
| | | | | | Capture the BLE traffic while interacting with the target device using Ubertooth One (Or with 2 BLE dongles with projects like BTLEJuice) | Relay based attacks | |
| | | | | | Write data to the target devices's BLE Characteristics using Gatttool | | |
| | | | | | Capture the initial pairing packets and use crackle to decrypt traffic (if encrypted) | | |
| | | | ZigBee | Find the ZigBee channel on which the DUT is operating on | Capture communication using zb_dump and analyze in wireshark | | |
| | | | | | Perform replay based attacks using zb_replay | | |
| | | | | | Identify keys in the captured communication | | |