PNNL-29518

Characterizing Vulnerabilities Associated with Connected Lighting

Exploring Authorization Protocols

December 2019

Paul Francik

In fulfillment of DOE Science Undergraduate Laboratory Internships (SULI) program requirements

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

PNNL-29518

Characterizing Vulnerabilities Associated with Connected Lighting

Exploring Authorization Protocols

December 2019

Paul Francik

In fulfillment of DOE Science Undergraduate Laboratory Internships (SULI) program requirements

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

Cities are upgrading their infrastructure and converting traditional indoor and outdoor luminaires to Connected Lighting Systems equipped with sensors in an aggressive effort to reduce energy consumption, increase sustainability, and improve the quality of life. When you take something seemingly benign like a lighting system and connect it to the internet, if improperly secured a potential attack vector for hackers is created. Hackers could then access sensitive information, pivot into other networks, shut down services, or enslave devices to do their bidding. In attempt to close this attack vector and secure these systems, this paper extends previous work that investigated, compared and contrasted authentication vulnerabilities in Connected Lighting Systems that identified the need for additional testing and improved test method documentation. One additional authentication test and four new authorization tests were developed utilizing the Open Web Application Security Project (OWASP) as a test development and documentation guide. The new tests were integrated with the previous work to develop an updated test method with a focus on describing test procedures to make them easy to understand and repeat. In addition, a first Connected Lighting System use case was established to begin developing threat profiles that will aid in the identification of vulnerabilities whose focus areas extend beyond authentication and authorization. This use case describes the generation and flow of CLS data which assists in specifying methods to harden CLSs from outside interference and exploitation that can be implemented immediately on existing CLSs and as new products come to market.

Acknowledgments

I would like to thank Benjamin Feagin Jr. [PNNL] for his mentorship, providing this internship opportunity with the Advanced Lighting Team in Portland Oregon.

I would also like to thank Michael Poplawski [PNNL] who provided guidance and insight that greatly assisted this research and the basis for characterizing cybersecurity vulnerabilities of connected lighting systems.

Thanks to Patrick O’Connell for his peer review of this paper,

Additional thanks to Kelly Gordon [PNNL], and Todd Samuel [PNNL] for their support of this project and its implementation.

Acronyms and Abbreviations

CLSs	Connected Lighting Systems
IOT	Internet of Things
LED	Light Emitting Diode
NIST	National Institute of Standards and Technology
PII	Personally Identifying Information
PHI	Protected Health Information
SDLC	Software Development Lifecycle
SI	Sensitive Information
URI	Uniform Resource Identifier
URL	Uniform Resource Locator (commonly referred to as a web address)

Contents

Figures

1.0 Introduction

To enhance energy savings, increase performance and interoperability, and reduce the impact on climate change, major cities are making the switch to connected lighting systems (CLSs).

With the emergence of light emitting diode (LEDs) technology, the opportunity to use 75% less energy consumption and have these products last approximately 25 times longer than an incandescent bulb is important to help decrease the load we put on our electrical grid and to reduce CO2 emissions. (energy.gov n.d.) According to Toby Morgan, LED program manager of The Climate Group “LED reach up to 80 percent savings when coupled with smart controls. This is unprecedented for a direct replacement technology.” (Theclimategroup.org 2019)

These findings are impressive, however if the goal is continued adoption of such systems across the globe it is important to understand the risks and vulnerabilities associated with any new technology or emerging platform. Further, it is of paramount importance to test these connected lighting devices to harden such systems from outside interference. One way to test systems that have come to market is to perform a risk assessment against known vulnerabilities and perform a penetration test associated with the different CLSs devices and configurations to determine if they pass well-known cyber security exploits.

A. Background on CLSs

CLSs luminaries are essentially a grouping of LED lights that are enhanced digital devices connected to the internet. It can be thought of like a basic computing device. When combined on a network together they form a connected lighting system. CLSs provide the ability to not only light a space, but with the right sensors attached can also collect data about traffic, pedestrians, weather, radiation, pollution, gunshot triangulation, emergency services and more. (lighting.philips.com 2019)

With the advancement of this technology and the implementation of these lighting devices on a network, they become part of the Internet of Things (IOT), an everyday object with computing devices that can send and receive data via the internet. CLSs are allowing cities to create an information highway of data that can be used to the cities benefit or if improperly secured, lead to catastrophic loss of service and breached networks.

2.0 Authentication

Authentication is a class or family of security controls that ultimately constitutes one or more processes to verify an identity claim. Defining the authentication process and needed mechanisms typically begins with an appropriate threat model where the definition of trust boundaries and identification of who or what may or may not traverse those boundaries in order to interact with an entity of value (e.g. a system, network, or specific data) occurs. Once authenticated, the subsequent actions an entity may perform is determined by authorization. Authorization is the process of verifying a requested action is permitted by assessing the privileges associated with or bound to the entity. Further, means must be developed for auditing actions and changes to both.

An identity is a unique representation of someone or something, such as a person or a node on a network. Common terms used to describe identities are users and machines. The most common authentication mechanism is the traditional username and password combination, also referred to as a credential set, with the password typically being a secret. Verification is typically performed by assessing one or more provided secret credentials against a stored representation or a stored copy of the secret. Successful authentication provides a reasonable assurance that an identity claimant is a legitimate entity and allowed to perform subsequent actions available only to authenticated entities. Humans and machines are both capable of using credential sets as well as other mechanisms such as cryptographic keys and tokens.

Once authenticated, an entity generally desires subsequent access to other local or remote entities (i.e., systems, networks, data, etc.). This subsequent access, referred to as authorization, is controlled by an access control policy that enumerates the permissions or rights associated with each entity and results in an access decision (i.e., allowed or denied) based on a set of rules. Policies may be simple, for example, a specific entity may access another specific entity, or complex, an entity with specific characteristics and conditions may access another entity with specific characteristics and conditions. The access control model utilized determines the simplicity or complexity of authorization and its implementation (e.g., through configuration) serves as another area of focus for security testing.

A. Authentication Vulnerabilities

Attackers looking to exploit authentication vulnerabilities target weaknesses in the communication medium used to transport secret credentials, the secrets storage mechanism, and/or in the authentication mechanism itself. Communication mediums, wired and wireless, may be susceptible to what are known as sniffing attacks where an attacker is able to observe secret credentials in an intelligible manner due to a lack of adequate cryptographic protections. Inadequacy may come in the form of a failure to implement available cryptographic protections, insecure (i.e., misconfigured) implementation, or weakness in the cryptographic mechanism itself (e.g., weak or broken ciphers). The same cryptographic concepts apply to secrets storage. If secret credentials are stored without cryptographic protections, an attacker can observe intelligible secrets within the storage mechanism.

Provided enough time and resources, an attacker could brute force any secret credential. A brute force attack against a secret credential may involve authentication attempts using all possible combinations that comprise the secret credential scheme (i.e., complexity requirements) and/or attempting to authenticate with credentials from one or more predefined dictionaries until a successful authentication occurs. A brute force attack may also involve attempts to reverse one or more cryptographic representations of the secret credential (e.g. cracking a password hash). The success of brute force attacks is largely a function of secret strength (i.e., length, character set, and/or cryptographic protections). Weaknesses in authentication can be identified and assessed using various methods such as static and/or dynamic code analysis, software composition analysis, vulnerability assessments, and penetration testing. Each method has its use cases and may be performed during development, post-release, by staff, or by independent third-parties.

Static and/or dynamic code analysis involves evaluating application source code and/or run time behavior for errors or conditions that may cause anomalous behavior. This is often done using automated tools but may also be performed manually using code reviews (e.g. pair programming or reviewing pull requests). Static and/or dynamic code analysis is often performed by development teams or security teams supporting development to catch as many potential vulnerabilities prior to release. An example may include the use of a debug option during development that when set would bypass authentication. Such an option is likely not meant to be included in a production release and static analysis, manual or automated, should catch this vulnerability. Software composition analysis is a method of attempting to determine software components used within an application and known vulnerabilities associated with those components. For example, software composition analysis is performed on an application and the analysis results reveal it uses a specific software library (i.e. version) vulnerable to an authentication bypass as reported in the National Vulnerability Database. Software composition analysis is typically an automated process that analyzes available source and binaries

3.0 Authorization

Authorization is the management of users' access to resources after being authenticated, sometimes referred to as access control. The primary function of an authorization management system is to control which users or processes can access certain objects and available files. The Access control model filters users into assigned roles and associated groups to efficiently manage authorization by applying privileges to these categories. These Privileges control what a user is authorized to do on the network. There are four main types of Authorization models that are often utilized in conjunction with one another to impose greater control and safeguards, but also serve as a means of accountability to assess which users and processes are accessing which objects.

Attribute Based Access Control (ABAC) uses Access Control Lists (ACLs) to validate attributes associated with specific users in alignment with the set of rules it has established. If the required attribute (name, employee ID, birth date) associated with that user and that object match up with what is on the ACL the user is authorized and granted access. If the user’s attributes don't match the rules of the ACL, access is denied.

Mandatory Access Control (MAC) is the strictest of all authorization models and is a rule-based system where confidentiality is a primary concern, therefore the rules must be implemented without exception. A primary example of this would be in the governmental sector related to security clearance levels. All objects are assigned a label associated with clearance level (top secret, secret, confidential, etc.) and given a category (department, work-group, manager, project). Each user on this system is also assigned a label and category affiliation that corresponds to the naming schema and rules set forth by the administrators. When a user attempts to access objects on the system, the MAC checks that both the category and label associated with the object matches the labels and categories associated with the user. If both criteria are met the user is authorized to access that particular object, if one or none of the criteria match then the user is denied access to that resource. These policies are set forth by the system administrator and individual users cannot change access labels or categories.

Discretionary Access Control (DAC) allows the owner of an object to assign permissions and privileges at their discretion. In this model each object or file is given an individual ACL with user names and work groups. Users or groups are allowed certain privileges as assigned by the owner of the object. This model is employed on many desktop operating systems. A user creates a file or document and then assigns what other users can read, modify, share etc.

Role Based Access Control (RBAC) and Group based Access Control (GBAC) assign attributes to a specific job position or role or group type rather than the individual user. A user must be assigned a certain job role to access the associated documents and systems with that role. A good example would be an individual is hired on as a human resource administrator. That user would then have access to all documents associated with the role of human resource administrator. In GBAC a user must be a member of a group to access the objects associated with that group attribute. The user assigned the role of human resource administrator would also get placed in the human resources group and have access to associated objects housed under that group. If a user doesn't belong to the group or job role as specified in the RBAC and GBAC they are denied permission to those objects.

A. Authorization Vulnerabilities

Often authorization vulnerabilities are a flaw located within the security architecture of the vulnerable system when authorization models are not applied consistently if at all. (CWE.mitre.org 2019) This type of vulnerability allows the user to access data or perform actions they otherwise should not be permitted to do. An example of how this flaw is introduced during implementation of new security architecture could be a developer not thinking like an aggressor and not having the knowledge that headers and cookies can be modified to gain access to objects. This can be a simple oversight or lack of knowledge in a developer's skill set. Further complicating the matter, if attributes or permissions aren't mapped out carefully in the RBAC model, inherited permissions could allow access that should otherwise be denied. Often these are tested initially during the development implementation before release, with ongoing testing throughout the development life cycle.

Authorization vulnerabilities are of major concern as they violate the CIA triad, a model designed to help implement security policies within organizations. CIA stands for Confidentiality, Integrity, and Availability. (Purcell 2018) Applied to this scenario Confidentiality would be the set of rules allowing access to the intended object. Integrity would apply to the data or objects being trustworthy and accurate without outside manipulation. Availability would relate to reliable access to those objects for authorized users. With big data comes big responsibility as often unauthorized access can lead to violations of personally identifying information (PII), Protected Health Information (PHI), or Sensitive Information (SI). Authorization vulnerabilities can incur catastrophic repercussions for companies and organizations whose sensitive or classified information is at risk as well as individuals whose PII, PHI, or SI is being compromised and used. These types of violations can lead to hefty financial penalties from governmental agencies and require mandatory compliance of a corrective action plan that will remediate such violations for the company or stakeholder who wasn't able to protect the information.

Attackers targeting vulnerabilities in authorization sometimes have already been authenticated with legitimate user credentials and are generally looking for a way to bypass the restrictions placed on that legitimate account in order to escalate privileges or gain access to objects they shouldn't be able to view. Through privilege escalation the attacker can increase their access on the compromised system and make changes and modifications to files and objects or other user permissions, leading to further compromise.

If web applications cache their pages, and don't use active secure session tokens to validate credentials, another way an attacker might evade authorization checks is simply through requesting direct access to that resource by typing in the specific destination or URL being used to cache that specific file or object. This can be avoided though the use of active and authenticated session tokens and making sure that all pages containing sensitive data are not cached and visible externally

4.0 Test Setup

Authentication test setup

The high-level setup that was established to perform the authentication tests is shown in Figure 1. In this setup, the tests are conducted through a user interface device that has both Windows and Kali Linux operating systems with at least one web browser installed. The user interface device must also have software installed for a login cracker, web vulnerability scanner, and a packet analyzer software. The user interface device must also be capable of performing over-the-air Zigbee hardware packet analyzer. The user interface is located on the private network connecting to a wireless gateway. On the private network is an Ethernet switch in which CLS are connected, a router, and a firewall. An example of how varying system architectures might be integrated into a network is shown in Figure 2, including a wireless system connected to a private network via a local gateway, a wireless system connected to a Public network via a shared gateway, a wired system that utilizes Ethernet or Power-over-Ethernet (PoE) based communication and can be configured using a mobile device connected via a Wi-Fi gateway, and a wireless system that utilizes Bluetooth Mesh both for communication and configuration via a mobile device.

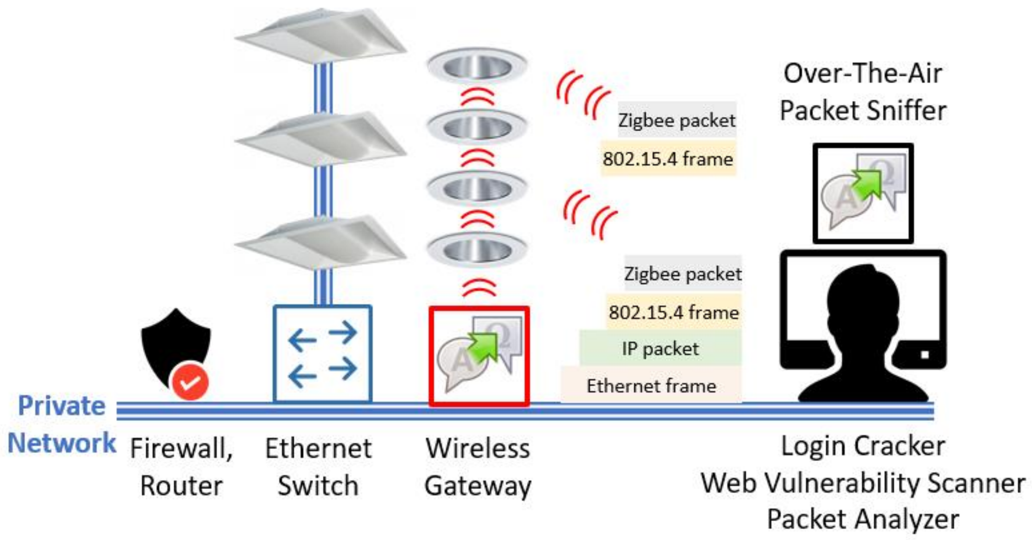


Figure 1. Authentication Test Setup

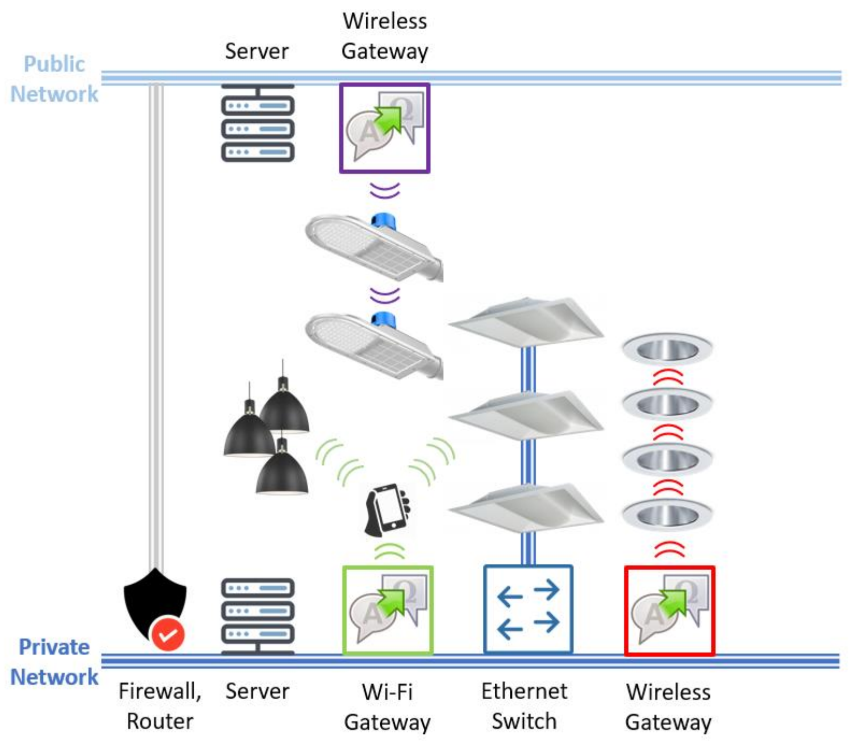


Figure 2. Varying System Architectures

Authorization Test Setup

The test setup used to identify authorization vulnerabilities consist of a user interface device with multiple operating systems, multiple web browsers, encoding and decoding software applications, directory traversal fuzzing software, string searcher software, multi-threaded java application designed to brute force directories and files names, an attack proxy, a java framework for analyzing applications that communicate using the HTTP and HTTPS protocols, and a web vulnerability scanner. On the private network is an Ethernet switch in which CLSs are to be connected, a router, and a firewall.

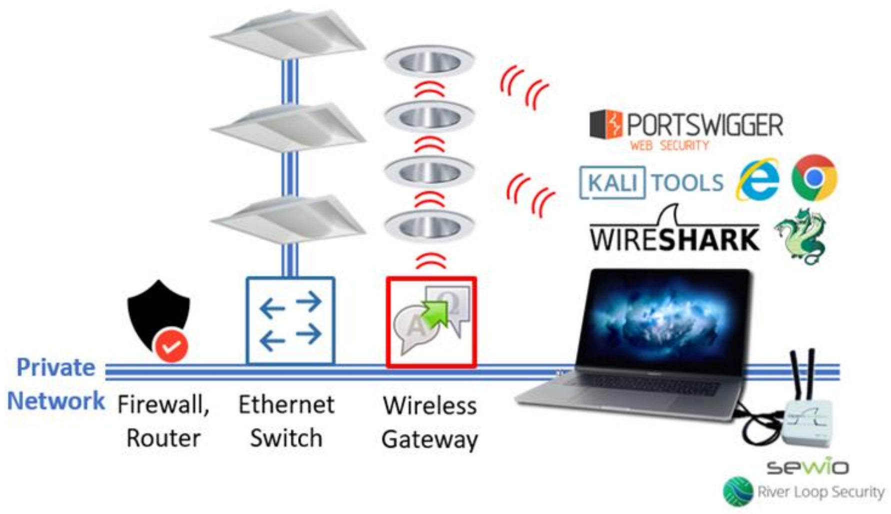


Figure 3. Authorization Test Setup

5.0 Authentication Tests Defined

A total of eighteen tests were developed by Underwriters Laboratories (UL) and implemented by PNNL in the connected lighting testbed to characterize the authentication vulnerabilities of Connected Lighting Systems (CLSs). The tests explore the implementation of basic authentication best practices (e.g., encrypting user credentials before transmitting them on the network) as well as known technology-specific (e.g., the use of Zigbee default trust center, or the implementation of JSON Web Token a.k.a. JWT replay protections) vulnerabilities.

Test	Type	Description
Test 1: Web Authentication Credentials Transported over an unencrypted Channel	Authentication	Determine whether authentication credentials (e.g., username, password) are encrypted (e.g., using HTTPS) prior to being transmitted on the network. Additional detail is provided in the OWASP Testing Guide[1].
Test 2: Use of Default Web Credentials	Authentication	Determine whether system has default accounts (e.g. admin) that, following installation, authenticate with a default account username/password. Additional detail is provided in the OWASP Testing Guide[2].
Test 3: Weak Web Lockout Mechanism	Authentication	Determine whether a user account is locked within 10 minutes after the initiation of a brute-force attack, or following 6 or more consecutive authentication failures. Additional detail is provided in the OWASP Testing Guide[3].
Test 4: Authentication Schema Bypass	Authentication	Determine whether authentication credentials that are included in the request header after a successful login are removed from the header after an authorized user logs out. Additional detail is provided in the OWASP Testing Guide[4].
Test 5: Insecure Authentication Credential Retention	Authentication	Determine whether session cookies store insecure (e.g., clear-text, unencrypted) authentication credentials. Additional detail is provided in the OWASP Testing Guide[5].
Test 6: Session Timeout	Authentication	Determine whether a user is automatically logged out from an active session following a period of inactivity of more than 15 minutes. Additional detail is provided in the OWASP Testing Guide[6].

Test	Type	Description
Test 7: Session Cookie Destruction	Authentication	Determine whether session cookies are properly destroyed upon de-authentication or session termination due to inactivity. Additional detail is provided in the OWASP Testing Guide[7] and elsewhere[8].
Test 8: Renewed Authentication for Lost or Terminated SSH Sessions over a Remote Interface	Authentication	Determine whether stored data from the previous SSH session can be used to bypass authentication mechanisms during a new session creation.
Test 9: Web Authentication Username Enumeration	Authentication	Determine whether authentication error messages disclose authorized usernames, thereby facilitating brute force attacks with known usernames. Additional detail is provided in the OWASP Testing Guide[9].
Test 10: Use of Zigbee Default Trust Center Link Key	Authentication	Determine whether the publicly known Zigbee default trust center link key is used to obtain keys.
Test 11: JSON Web Token (JWT) 'none' Algorithm Validation	Authentication	Determine whether a JSON Web Token (JWT) may be used to bypass validation by utilizing 'none' for the 'alg' field.
Test 12: Weak JSON Web Token (JWT) HMAC SHA256 Secret	Authentication	Determine whether the JSON Web Token (JWT) HMAC SHA256 secret can be obtained through a brute force attack.
Test 13: Missing JSON Web Token (JWT) 'jti', 'exp', and 'iat' Claims	Authentication	Determine whether JSON Web Token (JWT) replay protections have been implemented.
Test 14: Insecure Web-Based Credential Set Password Change	Authentication	Determine whether a current password is required during a password change procedure.
Test 15: Assuming User Identity Through SAML Login	Authentication	Determines whether an attacker can log in as a different user during SAML authentication using an XML library vulnerability.
Test 16: MQTT Authentication Credentials	Authentication	Determine whether the CONNECT packet sent from a MQTT client to a MQTT broker discloses authentication information.

Test	Type	Description
Test 17: Bluetooth Replay and On-The-Fly Data Modification	Authentication	Determine whether Bluetooth communication over a web interface can be exploited by a replay or modification of GATT operations, once the BLE_component is connected to the designated device.
Test 18: Identifying Bluetooth Class of Device/Service	Authentication	Determine whether Bluetooth devices broadcast Class of Device or Class of Service as part of their discovery beacons.

Figure 4. Authentication Tests Defined

6.0 Authorization Tests Defined

An additional set of 4 authorization tests and one authentication test were developed based off the OWASP testing guide as a test development and documentation guide to explore known authorization vulnerabilities.

Test	Type	Description
TEST 1: Directory Traversal	Authorization	Determine whether access control can be circumvented through manual manipulation or software analysis of hidden directories and determine if files and directories can be read and accessed without proper authorization.
TEST 2: Bypass Authentication Schema	Authorization	Determine whether the implementation of the authorization schema for all roles restricts access to resources when not authenticated.
TEST 3: Privilege Escalation	Authorization	Determine whether a user can elevate their privileges to gain access to resources above their assigned role or function.
TEST 4: Insecure Direct Object Reference	Authorization	Determine whether authorization protocols and access control lists allow for direct supplied user input that allow or deny the retrieval of an object or file.

Figure 5. Authorization test Defined

Test 19: Bypass Authentication with SQL Injection	Authentication	Determine if a prepared SQL statement can bypass Authentication to gain login access

Figure 6. Additional Authentication Test Defined

7.0 Authentication testing guide

A. Web Authentication Credentials Transported over an Unencrypted Channel

Educational Description

Determine if your username and password are hidden from other people on the network. By tracking the login process, you can check for credentials in the data that external parties would be able to see.

Technical Description

This test uses a web proxy to log the authentication process and can determine if the transport mechanism that is used to exchange web authentication credential sets between clients and servers is protected using encrypted communication.

Required Tools

- Test system (e.g. general computing laptop, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)
- Web proxy (e.g. Burp, ZAP, etc.)

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network
3. Connect test system to the network
4. Launch and configure the web proxy on the test system to proxy HTTP/HTTPS request traffic
5. Launch and configure the web browser to use the web proxy
6. Validate that web traffic is being proxied in the web proxy

Testing

1. Navigate to the DUT Login Page and supply authentication credentials
2. Examine the authentication request to the DUT Login Page in the web proxy and note the request URI protocol used

Evaluation

This test is failed if the credentials are discovered in the web proxy.

B. Use of Default Web Credentials

Educational Description

Determine whether system has default accounts (e.g. admin) that, following installation, authenticate with a default account username/password.

Technical Description

This test determines if default web authentication credentials exist.

Required Tools

- Test system (e.g. general computing laptop, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.) Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)
- Web proxy (e.g. Burp, ZAP, etc.)
- THC-Hydra (installed on test system)

Steps

1. Install THC-Hydra on the test system (if applicable)
2. Establish a local private network using the network infrastructure
3. Connect DUT to the network
4. Connect test system to the network
5. Launch and configure the web proxy on the test system to proxy HTTP/HTTPS request traffic
6. Launch and configure the web browser to use the web proxy
7. Validate that web traffic is being proxied in the web proxy

Testing

1. Navigate to the DUT Login Page and supply authentication credentials
2. Examine the authentication request to the DUT Login Page in the web proxy and note the request method, request URI, and the parameters and arguments used
3. Configure and invoke THC-Hydra in the following manner:
   hydra -L usernames -P passwords [DUT Web IP Address] http-post-form "[DUT HTTP POST Form URI]:User=^USER^&Password=^PASS^:Login failed" -V -f

Evaluation

This test is failed by identifying default web credentials.

C. Weak Web Lockout Mechanism

Educational Description

This test determines if an account lockout mechanism is used and if a sufficient time delay has been implemented.

Technical Description

Determine whether a user account is locked within 10 minutes after the initiation of a brute-force attack, or following 6 or more consecutive authentication failures.

Required Tools

- Test system (e.g. general computing laptop, workstation, etc.)

- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)
- Timer

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network
3. Connect test system to the network
4. Launch the web browser

Testing

1. Navigate to the DUT Login Page and supply known bad or complex random authentication credentials (perform 6 times or until an account lockout notification is presented)
2. Navigate to the DUT Login Page and supply known good authentication credentials (perform once every 5 minutes until successfully authenticated or 15 minutes has elapsed)

Evaluation

This test is failed by identifying a weak or absent web account lockout mechanism.

D. Authentication Schema Bypass

Educational Description

Determine whether authentication credentials that are included in the request header after a successful login are removed from the header after an authorized user logs out.

Technical Description

This test determines if an authentication schema bypass exists by directly accessing pages requiring prior authentication.

Required Tools

- Test system (e.g. general computing Linux laptop, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)
- Web proxy (e.g. Burp, ZAP, etc.)

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network
3. Connect test system to the network
4. Launch and configure the web proxy on the test system to proxy HTTP/HTTPS request traffic
5. Launch and configure the web browser to use the web proxy
6. Validate that web traffic is being proxied in the web proxy

Testing

1. Identify the authentication URI and parameters
2. Identify authentication (e.g. "token" if not present in the URI)
3. Re-using authentication parameter/token/value after user logged out, execute the following bash script:
   for URL in `cat ${filename_output_spider}`; do echo $URL curl -v -m 5 -s -I $1 "$URL" 2>&1 | grep HTTP/${http_version} | grep ${authenticated_message} done
4. Verify URIs associated with an authenticated portion of the web site returned an Authentication header with the following terminal command
   cat ${filename_output_spider} | stdbuf -oL grep -rl '${authenticated_message}' * | head -n1

Evaluation

This test is failed by verifying that some of the URLS belonging to authenticated area returned an Authentication header.

E. Insecure Authentication Credential Retention

Educational Description

Determine whether session cookies store insecure (e.g., clear-text, unencrypted) authentication credentials.

Technical Description

This test determines if the 'remember me' feature involved with web authentication is implemented insecurely.

Required Tools

- Test system (e.g. general computing Linux laptop, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)
- Web proxy (e.g. Burp, ZAP, etc.)

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network
3. Connect test system to the network
4. Launch and configure the web proxy on the test system to proxy HTTP/HTTPS request and response traffic
5. Launch and configure the web browser to use the web proxy
6. Delete all previous web browsing history, data, cookies, saved passwords, forms, etc.
7. Validate that web traffic is being proxied in the web proxy

Testing

1. Navigate to the DUT Login Page, supply authentication credentials, and ensure the 'remember me' feature is enabled
2. Examine the response in the web proxy and note the cookie value
3. Logout and terminate the web browser
4. Launch the web browser and navigate to the DUT Login Page
5. Examine the request in the web proxy and note the cookie value

Evaluation

This test is failed by discovering the password in the cookie.

F. Session Timeout

Educational Description

Determine whether a user is automatically logged out from an active session following a period of inactivity of more than 15 minutes.

Technical Description

This test determines if an inactivity timeout implemented in the authenticated area of the web application exists and is implemented securely.

Required Tools

- Test system (e.g. general computing Linux laptop, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, Internet Explorer)
- Burp Suite
- Burp Suite extension: Session Timeout (TestBApp Store > Extension name: Session Timeout Test)

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network infrastructure
3. Connect testing system to the network infrastructure
4. Launch and configure Burp Suite on the test system to proxy HTTP/HTTPS request and response traffic
5. Launch and configure the network adapter settings of the DUT to route traffic to use the web proxy
6. Log into the web interface of the DUT and navigate to site pages
7. Verify that the web traffic is being proxied in Burp Suite
8. Within Burp Suite, select Burp Extender
9. Within Burp Suite and with the Extender tab selected, select BApp Store
10. Search and install "Session Timeout Test" extension

Testing

1. Navigate to the DUT Login Page and login with valid authentication credentials

2. Examine the response returned by the DUT for the request involving the DUT Login Page in the Site Map or HTTP History of Burp Suite
3. Record a string (e.g. specific markup text) from the response that uniquely identifies the response as the DUT Login Page (e.g.)
4. Right-click on the request in the Site Map or HTTP History of Burp Suite and select "Test for Session Timeout"
5. Enter the string that will indicate the DUT response requires authentication, set the minimum and the maximum session duration, and the testing interval
6. The extension issues requests to the DUT based on the durations and interval configured until a response containing the string is detected

Evaluation

This test is failed if no session timeout occurs and the DUT has not implemented session timeouts.

G. Session Cookie Destruction

Educational Description

Determine whether session cookies are properly destroyed upon deauthentication or session termination due to inactivity.

Technical Description

This test examines if session cookies are properly destroyed upon deauthentication or session termination.

Required Tools

- Test system (e.g. general computing Linux laptop, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)
- Burp Suite

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network infrastructure
3. Connect test system to the network infrastructure
4. Launch and configure the web proxy on the test system to proxy HTTP/HTTPS request and response traffic
5. Launch and configure the network adapter settings of the DUT to use the web proxy
6. Log into the interface of the DUT and navigate to pages within its web portal
7. Verify that web traffic is being proxied in the web proxy

Testing

1. With Burp Suite proxy intercept turned off, navigate to the login page of the DUT's application
2. Log into the DUT's application using known good authentication credentials
3. Turn the intercept option 'on' in Burp Suite and refresh the authentication page to ensure only current session data is available
4. Right-click on the request and select "Send to Repeater"
5. Within Burp Suite select the Repeater tab and record the cookie value from the 'params' tab of the response
6. Disable Burp Suite's intercept and invoke the logout function of the DUT's application
7. Use the web browser's back button to attempt to return to a previously authenticated page
8. Repeat steps 1-5 and record the cookie value for comparison
9. Restore the old cookie value by replacing the current cookie and attempt to navigate to an authenticated page

Evaluation

1. This test is failed if after using the web browser's back button, the user is still able to view or interact with the DUT.
2. This test is failed if after invoking the DUT's logout function, the cookie value doesn't change.

H. Renewed Authentication for Lost or Terminated SSH Sessions over a Remote Interface

Educational Description

Determine whether stored data from the previous SSH session can be used to bypass authentication mechanisms during a new session creation.

Technical Description

This test checks if stored data from the previous SSH session is used to bypass authentication/authorization mechanisms during a new session creation.

Required Tools

- Test system (e.g. general computing Linux laptop)
- Network infrastructure (e.g. switch, hub, etc.)
- Terminal (Linux), putty.exe (Windows), Arping/Netdiscover

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network infrastructure
3. Connect test system to the network infrastructure

Testing

1. Using a terminal, establish an SSH connection using the command, and supply the password for the username when prompted:
   ssh $(login_usr)@$(dut_ipaddr)
2. Unplug the network cable from the DUT
3. Wait one minute and reconnect the network cable to the DUT
4. Observe the state of the established SSH session in the terminal

Evaluation

This test is failed by the DUT not requiring renewed authentication after the network cable has been disconnected from the DUT.

I. Web Authentication Username Enumeration

Educational Description

Determine whether authentication error messages disclose authorized usernames, thereby facilitating brute force attacks with known usernames.

Technical Description

This test determines if authentication error messages disclose the existence of usernames which may allow for username enumeration or account harvesting.

Required Tools

- Test system (e.g. general computing device, workstation, etc.)
- Network infrastructure (e.g. switch, hub, etc.)
- Web browser (e.g. Chrome, Firefox, Safari, IE, Edge)

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT to the network infrastructure
3. Connect test system to the network infrastructure
4. Launch the web browser on the test system
5. Clear the web browser's previous web browsing history, data, cookies, saved passwords, forms, etc.
6. Using the web browser, navigate to the DUT's authentication page

Testing

Provide authentication credentials using a known valid user name and an invalid password
If a known valid username is unavailable, attempt to provide common usernames with random passwords (e.g. admin, administrator, root, support, operator, technician, developer, etc.)

Evaluation

This test is failed if returned error messages disclose authentication information such as "Incorrect password".

J. Use of Zigbee Default Trust Center Link Key

Educational Description

Determine whether the publicly known Zigbee default trust center link key is used to obtain keys.

Technical Description

This test determines if the publicly known Zigbee default trust center link key is used.

Required Tools

- Test system (general purpose Linux computing system)
- Zigbee enabled device
- Atmel RZRAVEN flashed with Killerbee (Killerbee dongle)
- Wireshark

Steps

1. Insert the Killerbee dongle into the general-purpose Linux computing test system
2. Power on the DUT and an appropriate Zigbee enabled device
3. Launch a terminal on the test system and execute the command "zbstumbler -v" to determine the Zigbee channel used by the DUT
4. Record the Zigbee channel used by the DUT
5. Launch a terminal on the test system and execute the command "zbwireshark -c ${Zigbee channel}
6. Configure Wireshark Zigbee ("ZigBee") protocol preferences for the "AES-128, 32-bit MIC" Security Level to use key value "5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39" (Normal byte order)
7. Initiate a Zigbee pairing between the DUT and the Zigbee enabled device

Testing

Determine from the captured Zigbee traffic if a Transport Key frame (0x05) is observable. This Transport Key frame will contain a network key (e.g. "08:93:10:26:20:4c:22:98:2d:20:33:50:04:a0:61:50")

Evaluation

This test is failed by observing Transport Key frames in the captured Zigbee traffic.

K. JSON Web Token (JWT) 'none' Algorithm Validation

Educational Description

Determine whether a JSON Web Token (JWT) may be used to bypass validation by utilizing 'none' for the 'alg' field.

Technical Description

This test determines if a JSON Web Token (JWT) may bypass validation by utilizing 'none' for the 'alg' field.

Required Tools

- Test system (general purpose Linux computing system)
- curl
- Wireshark
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- JWT Debugger (https://www.jwt.io)

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT (API endpoint) to the network infrastructure
3. Connect test system to the network infrastructure
4. Launch Wireshark and begin capturing network traffic
5. Launch a terminal to be used with curl
6. Launch a web browser and navigate to the Debugger available at https://www.jwt.io

Testing

1. Generate a successful authentication to the DUT or determine the fields used by the JWT from API documentation
2. Record the returned JWT or manually generate a JWT that adheres to the format expected by the DUT
3. If a JWT was captured, decode it using the web browser and https://jwt.io
4. Manipulate the JWT header to use 'none' as the value for the 'alg' field
5. Manipulate the JWT payload to include appropriate fields and values (e.g. "loggedInAs":"admin")
6. Encode the modified JWT ensuring the signature portion is removed and ends with a . (dot)
7. Transmit the modified JWT to the DUT to interact with a resource using curl
   e.g. **curl -X POST -d "user=admin&action=delete" http://${api-endpoint} -H "Authorization: JWT ${jwt}"**

Evaluation

This test is failed if the DUT takes an action on a resource with a modified JWT containing a header with the 'alg' field set to 'none'. This indicates validation is not performed correctly.

L. Weak JSON Web Token (JWT) HMAC SHA256 Secret

Educational Description

Determine whether the JSON Web Token (JWT) HMAC SHA256 secret can be obtained through a brute force attack.

Technical Description

This test attempts to brute force the JSON Web Token (JWT) HMAC SHA256 secret.

Required Tools

- Test system (general purpose Linux computing system)
- curl
- Wireshark
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- https://jwt.io Debugger

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT (API endpoint) to the network infrastructure
3. Connect test system to the network infrastructure
4. Launch Wireshark and begin capturing network traffic
5. Launch a terminal to be used with curl
6. Launch a web browser and navigate to the Debugger available at https://jwt.io

Testing

1. Generate a successful authentication to the DUT or determine the fields used by the JWT from API documentation
2. Record the returned JWT or manually generate a JWT that adheres to the format expected by the DUT
3. If a JWT was captured, decode it using the web browser and https://jwt.io
4. Generate a valid JWT (header and payload) using the data obtained during the decoding step
5. Manipulate the JWT signature secret value
6. Encode the modified JWT
7. Transmit the modified JWT to the DUT to interact with a resource using curl
   e.g. **curl -X POST -d "user=admin&action=delete" http://${api-endpoint} -H "Authorization: JWT ${jwt}"**

Evaluation

1. Repeat the process of manipulating the JWT signature secret value until secret values have been exhausted (e.g. from a dictionary) or the requested action on the resource completes successfully
2. This test is failed if the DUT takes an action on a resource for all supplied JWT signature secret values. This indicates a weak HMAC SHA256 secret was used.

M. Missing JSON Web Token (JWT) 'jti', 'exp', and 'iat' Claims

Educational Description

Determine whether JSON Web Token (JWT) replay protections have been implemented.

Technical Description

This test determines if JSON Web Token (JWT) replay protections have been implemented.

Required Tools

- Test system (general purpose Linux computing system)
- Wireshark
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- https://jwt.io Debugger

Steps

1. Establish a local private network using the network infrastructure
2. Connect DUT (API endpoint) to the network infrastructure
3. Connect test system to the network infrastructure
4. Launch Wireshark and begin capturing network traffic
5. Launch a web browser and navigate to the Debugger available at https://jwt.io

Testing

1. Generate a successful authentication to the DUT
2. Record the returned JWT 3. Decode the recorded JWT using the web browser and https://jwt.io

Evaluation

This test is failed if 'iat' (issued at), 'exp' (expiration timestamp), and 'jti' (nonce) claims do not exist in the JWT payload. If these claims are absent, JWT replay attacks may be possible.

N. Insecure Web-Based Credential Set Password Change

Educational Description

Determine whether a current password is required during a password change procedure.

Technical Description

This test determines if a current password is required during a password change procedure.

Required Tools

- Test system (general purpose Linux computing system)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)

Steps

1. Establish a local private network using the network infrastructure
2. Connect the test system to the network infrastructure
3. Launch the web browser on the test system
4. Clear the web browser's previous web browsing history, data, cookies, saved passwords, forms, etc.
5. Using the web browser, navigate to the DUT's authentication page

6. Provide authentication credentials using a known valid user name and an invalid password

Testing

Navigate to DUT's password change page

Evaluation

This test is failed if the current password is not required to change the password.

O. Assuming User Identity Through SAML Login

Educational Description

Determines whether an attacker can log in as a different user during SAML authentication using an XML library vulnerability.

Technical Description

This test determines if an attacker can log in as a different user during SAML authentication using a vulnerability in XML libraries.

Required Tools

- Test system (general purpose Linux computing system)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- Burp Suite
- SAML Raider - Burp Suite extension

Steps

1. Open Burp Suite and click Extender tab and then BApp Store
2. In the Burp extensions list find SAML Raider, click on it, then click Install button
3. Launch and configure the web proxy on the test system to proxy HTTP/HTTPS request and response traffic
4. Establish a local private network using the network infrastructure
5. Connect the test system to the network infrastructure
6. Launch the web browser on the test system
7. Clear the web browser's previous web browsing history, data, cookies, saved passwords, forms, etc.
8. Using the web browser, navigate to the DUT's authentication page
9. Create a new account with an email similar to existing user email, e.g. if user's email is admin@mywebsite.com, a testing email will be admin@mywebsite.com.fakeweb.com
10. Logout DUT's user

Testing

1. Using the web browser, navigate to the DUT's authentication page
2. Launch Burp Suite, click on Proxy tab and press Intercept button
3. In the browser provide authentication credentials using a newly created user

4. After successful sign-on modify Idp response in Saml Raider tab
5. Find NameId attribute and insert comment before fakeweb.com e.g. admin@mywebsite.com.fakeweb.com admin@mywebsite.com.fakeweb.com
6. Click Forward button

Evaluation

This test is failed if you can login as admin@mywebsite.com. If yes, it means that your Service Provider is not setup correctly, or use outdated XML packages

P. MQTT Authentication Credentials

Educational Description

Determine whether the CONNECT packet sent from a MQTT client to a MQTT broker discloses authentication information.

Technical Description

This test checks if the CONNECT packet from a MQTT Client to a MQTT broker discloses authentication information.

Required Tools

- Test System (General computing Windows laptop)
- Network infrastructure (e.g. switch, hub, etc.)
- Wireshark

Steps

1. Establish a local private network using the network infrastructure i.e. switch
2. Connect DUT (MQTT Client) to the network
3. Configure and connect DUT (MQTT Broker) on the Testing System to the network
4. Configure Wireshark to capture traffic on the local interface on the Testing System

Testing

1. Register the DUT (MQTT Client) device to the DUT (MQTT broker) using the web interface of the DUT (MQTT Client)
2. Wireshark now captures this CONNECT packet from the DUT (MQTT Client) which is the packet of Interest for the test
3. A Wireshark packet filter "tcp.port == 8883 AND ip.addr "DUT (MQTT Client)" is used to inspect the connect message
4. View the protocol section on Wireshark of the captured packet, identify the username and password fields

Evaluation

This test is failed if in the event, the DUT (MQTT Client) credentials are in clear text; this gives the attacker an opportunity to implement a MITM attack

Q. Bluetooth Replay and On-The-Fly Data Modification

Educational Description

Determine whether Bluetooth communication over a web interface can be exploited by a replay or modification of GATT operations, once the BLE_component is connected to the designated device.

Technical Description

This test exploits Bluetooth communication over the web interface once the BLE_component is connected to the designated device.

Required Tools

- Two Testing Systems (General computing Kali Linux laptop)
- Network infrastructure (e.g. switch, hub, etc.)
- BTLEjuice Bluetooth Adapter (Generic/In-Built that support v4 Bluetooth)
- Two Bluetooth DUTs (Master and Slave)

Steps

1. Establish a local private network using the network infrastructure i.e. switch
2. Connect the Bluetooth Adapter to the testing systems and make sure the adapter is available from the system
3. Connect Testing System 1 with "btlejuice framework" installed to the switch and launch the btlejuice-proxy using the command "sudo btlejuice-proxy"
4. Connect Testing System 2 to the switch and run the following command "sudo btlejuice -u 'proxy(Testing System 1) IP Address' -w"
5. The web interface is now available at "http:/localhost:8080" on the Testing System 2

Testing

1. Using the web Interface on the Testing Laptop 2, a target DUT(Master) is selected from the list of all available Bluetooth Low Energy (BLE) devices detected by the intercepting core
2. Once the interface is ready, Testing Laptop 2 typically clones/duplicates the DUT, thus forming a dummy device that pretends to be the DUT(Master), use the associated BLE DUT(Slave) to connect to the dummy device
3. All the intercepted GATT operations are then displayed with the corresponding services and characteristics UUID, and the data associated with them
4. Replay GATT operations: A GATT operation can be replayed by right-clicking it and then selecting write (or Read) button to replay the corresponding GATT operation.
5. On-the fly data modification: A service/characteristic can be hooked and modified to change the BLE device operation

Evaluation

This test is failed if the DUT (slave) responds to the replay/modification of GATT operations. This indicates that the communication is not encrypted and/or a mechanism to sign the data with a CSRK is not implemented

R. Identifying Bluetooth Class of Device/Service

Educational Description

Discover if the device is sending identifying information in signals that are constantly being sent to all nearby Bluetooth devices.

Technical Description

Determine whether Bluetooth devices broadcast either Class of Device or Class of Service as part of their discovery beacons.

Required Tools

- Testing System (General computing Kali Linux laptop)
- hcitool
- Bluetooth Adapter (Generic/In-Built)

Steps

1. Connect the Bluetooth Adapter to the Testing Laptop
2. The following commands use hciconfig to enable your Bluetooth Adapter. Run the following commands:
a. hciconfig
b. hciconfig hci0 up

Testing

1. Scan for Bluetooth devices with hcitool on kali.
Note: Ensure the DUT is sending out discovery beacons (i.e. turn on discovery mode). Run the following commands:
a. hcitool scan
b. hcitool inq
2. Note that the "hcitool scan" gives out the MAC address of the DUT, and the "hcitool inq" command displays the clock offset and the class of device (CoD)
3. Convert the hex CoD to Binary and split it in the following fashion:
a. Bits 0-1: Format Type
b. Bits 2-7: Minor Device Class
c. Bits 8-12: Major Device Class
d. Bits 13-23: Major Service Class
4. These bits can be broken down to see what Bluetooth services DUT has configured
a. Bluetooth CoD: Format Type This is a 2-bit mask so there are 4 possible values 00,01,10,11

b. Bits 2-7: Minor Device Class It should be noted that it's possible to have more than one Minor Device Class enabled but when this is the case the Major Device Class reported should be similar to the primary Minor Device Class. **Note**: For each Major Device Class there are a possible 64 Minor Device Class combinations as calculated by 6 possible bits.

c. 3. Bits 8-12: Major Device Class, the current Major Device Class Values are, Miscellaneous, Computer, Phone, LAN/Network Access Point, Peripheral, Imaging, Wearable, Toy, Health, Uncategorized, and Reserved.

d. 4. Bits 13-23: Major Service Class CoD Major Service Class
Bit 13: Limited Discoverable Mode CoD Major Service Class
Bit 14: (reserved) CoD Major Service Class
Bit 15: (reserved) CoD Major Service Class
Bit 16: Positioning (Location identification) CoD Major Service Class
Bit 17: Networking (LAN, Ad hoc, ...) CoD Major Service Class
Bit 18: Rendering (Printing, Speaker, ...) CoD Major Service Class
Bit 19: Capturing (Scanner, Microphone, ...) CoD Major Service Class
Bit 20: Object Transfer (v-Inbox, v-Folder, ...) CoD Major Service Class
Bit 21: Audio (Speaker, Microphone, Headset service, ...) CoD Major Service Class
Bit 22: Telephony (Cordless telephony, Modem, Headset service, ...) CoD Major Service Class
Bit 23: Information (WEB-server, WAP-server, ...)

Evaluation

This test is failed if information collected in this phase is critical to covertly discovering and collecting information about a target system.

S. Bypass Authentication with SQL Injection

Educational Description

Standard Query Language (SQL) Injection is a malicious attack where injected code interacts with the back end of the database to display content that should not be viewable. By using special characters that are used in the database configuration, errors may be produced which indicate how to craft special statements that override authentication credentials. This test checks whether these special characters are allowed.

Technical Description

Perform SQL Injection on the DUT authentication page to test if the code is executed.

Required Tools

- Testing System (General computing Kali Linux laptop)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- Web Proxy (OWASP ZAP, Burp Suite etc.)

Steps

1. Establish a local private network using the network infrastructure

2. Connect the test system to the network infrastructure
3. Launch the web browser on the test system
4. Using the web browser, navigate to the DUT's authentication page

Testing

1. Enter a single ' in the user name login field and hit enter
2. Look to see if the query is displayed and how it is formatted
3. Craft a SQL statement that matches the formatting of the query and enter it into the username field. (For example: ' or 1=1 –) Hit enter to see if you gain access
4. logout to the home screen
5. In the login field enter the name of a known user
6. In the password field enter a single ' and hit enter
7. Look at how the query is crafted, and reenter login credentials based off the set parameters. For example: login: user1 Password: ' or 1=1 –
8. Craft another statement if the other attempts were not successful. For example; login: user1 Password: ' or 1=1 and password = 'user1' –
   login: user1 Password: 1=1' or pass123 You could also try Username: admin'-- Password: SELECT * FROM members WHERE username = 'admin'--' AND password = 'password'
9. Other common injections into the login field to bypass authentication could be

- admin' --
- admin' #
- admin'/*
- ' or 1=1--
- ' or 1=1#
- ' or 1=1/*
- ') or '1'='1--
- ') or ('1'='1--

Evaluation:

This test is failed if a successful login happens bypassing Authentication and Authorization protocols due to SQL injection or if a database can be dumped to reveal user names and passwords or other sensitive and identifying information.

reference: https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

8.0 Authorization Testing Guide

A. Directory Traversal

Educational Description

Web servers and applications manage files into physical directories on file systems as part of their daily operation. When designed poorly, these systems and files can be accessed by an aggressor using inputs that exploit these vulnerabilities to gain access to files and data that should not otherwise be accessible.

Technical Description

Path Traversal tests are conducted through the use of input vectors enumeration and methodical testing techniques. This test determines whether an attacker is able to circumvent Access Control Lists (ACL) with the ability to read directories or files which they normally should not be able to access i.e. "root directory or /etc/passwd"

Required Tools:

- Testing System (General computing Kali Linux laptop)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- Web Proxy (OWASP ZAP, Burp Suite etc.)
- Encoding / Decoding tools
- Directory Traversal Fuzzer (e.g. DotDotPwn)

(owasp.org 2016)

Steps

1. Establish a local private network using the network infrastructure
2. Connect the test system to the network infrastructure
3. Launch the web browser on the test system
4. Using the web browser, navigate to the DUT's authentication page
5. Log in and authenticate with user credentials.

Testing

1. Check for request parameters that could be used for file related operations
2. Check for unusual variable names or file extensions that can be manipulated
3. Open and run DotDotPwn Directory Traversal fuzzer for web applications in the testing system laptop and document findings
   ```
   ./dotdotpwn.pl -m http -h 192.168.xxx.xx
   ```
4. Open OWASP ZAP and begin to proxy the web traffic through the appropriate browser. With necessary permissions spider the web application and document findings.

Evaluation

This test is failed if information collected in this phase is critical to covertly discovering and collecting information about a target system, hidden directories, files, or objects that contain sensitive information. Results from Dotdotpwn will show as "vulnerable" and OWASP ZAP will give a risk rating of low, medium, or high priority depending on the vulnerabilities discovered in this phase.

B. Bypass Authentication Schema

Educational Description

Users are assigned roles that are mapped to certain privileges and restrictions. A bypass flaw allows aggressors to circumvent or go around security mechanisms without authenticating and gain access to networks and resources.

Technical Description

This test checks the implementation of the authorization schema for all roles and whether access to resources is allowed when not authenticated.

Required Tools:

- Testing System (General computing Kali Linux laptop)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- Web proxy (OWASP ZAP, Burp Suite, etc.)

(owasp.org 2014)

Steps

1. Establish a local private network using the network infrastructure
2. Connect the test system to the network infrastructure
3. Launch the web browser on the test system
4. Using the web browser, navigate to the DUT's authentication page
5. Log in and authenticate with Admin user credentials

Testing

1. Check for all privileged administrative functions such as "add new user" "delete user" etc. and see if they are accessible through direct URL page request
2. Log in as unprivileged user 2 and check if you can access those same resources through direct URL request as above (forced browsing)
3. Log out all users and see if those resources can be accessed without Authentication through URL request or if it is denied
4. Launch OWASP ZAP and begin to proxy internet traffic
5. Check when logging into the DUT application if it verifies a successful log in on the basis of a fixed value parameters
6. If fixed value parameters exist log out and attempt to manipulate through parameter modification

7. Check for any shared drives or shared resources and whether assigned roles can access objects they shouldn't have access to directly or indirectly

Evaluation

This test is failed if system manipulation results in collecting information or accessing resources that should be prevented by the Access Control Model.

C. Privilege Escalation

Educational Description

When a user modifies their privileges or gains access to resources or functionality above, or laterally outside their traditional role and it should have been prevented by a security mechanism, they are said to have achieved privilege escalation.

Technical Description

This test determines whether vertical or horizontal privilege escalation is possible.

Vertical escalation: (access to privileged accounts)

Horizontal escalation (access to similarly configured accounts)

Required Tools:

- Testing System (General computing Kali Linux laptop)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- Web proxy (OWASP ZAP, Burp Suite, etc.)

(owasp.org 2017)

Steps

1. Establish a local private network using the network infrastructure
2. Connect the test system to the network infrastructure
3. Configure Browser to use OWASP ZAP as a proxy
4. Launch the web browser on the test system
5. Using the web browser, navigate to the DUT's authentication page
6. Log in and authenticate with user credentials

Testing

1. Explore the application manually looking for the ability to manipulate user groups
2. Explore the application manually looking for the ability to manipulate user profiles
3. Explore the application manually looking for the ability to manipulate condition values
4. Explore the application manually looking for the ability to manipulate IP values to get around IP source identification methods
5. Look at issues the OWASP ZAP passive scanner has discovered

6. Use the active OWASP ZAP active scanner with appropriate permissions to scan for vulnerabilities not seen through manual inspection

Evaluation

This test is failed if privilege escalation is possible.

D. Insecure Direct Object References

Educational Description

Through the use of changing parameter values of user supplied input that point to direct objects, aggressors can bypass authorization mechanisms and gain access to databases, file systems and other useful information that should not be allowed.

Technical Description

This test takes user supplied input and checks to see if authorization protocols allow or deny the retrieval of an object or file through direct user input.

Required Tools:

- Testing System (General computing Kali Linux laptop)
- Web browser (e.g. Chrome, Firefox, Safari, etc.)
- Web proxy (OWASP ZAP, Burp Suite, etc.)
- Two or more users (easier to assess and verify objects owned and referenced by another user)

(owasp.org 2014)

Steps

1. Map out all locations in the application where user input is used to reference objects directly
2. Establish a local private network using the network infrastructure
3. Connect the test system to the network infrastructure
4. Log into the web interface of the DUT and navigate to site pages and map out all locations in the application where user input is used to reference objects directly

Testing

1. Attempt to modify the value of the parameters directly in the URL used to reference objects and determine whether it is possible to retrieve objects belonging to other users bypassing authorization
2. Have two users signed in simultaneously for testing, one with administrative privileges and one unprivileged user.
3. Have the administrative user create a new object, and see if the unprivileged user can manipulate application functionality to gain access to that object through direct object reference

4. Launch and configure the network adapter settings of the DUT to route traffic to use the web proxy
5. Log into the web interface of the DUT and navigate to site pages
6. Verify that the web traffic is being proxied
7. Inspect GET requests for previously mapped out locations and see if manipulation leads to an insecure direct object reference.

Evaluation

This test is failed if the retrieval of an object or file through direct user input can be obtained.

9.0 Threat Profiles

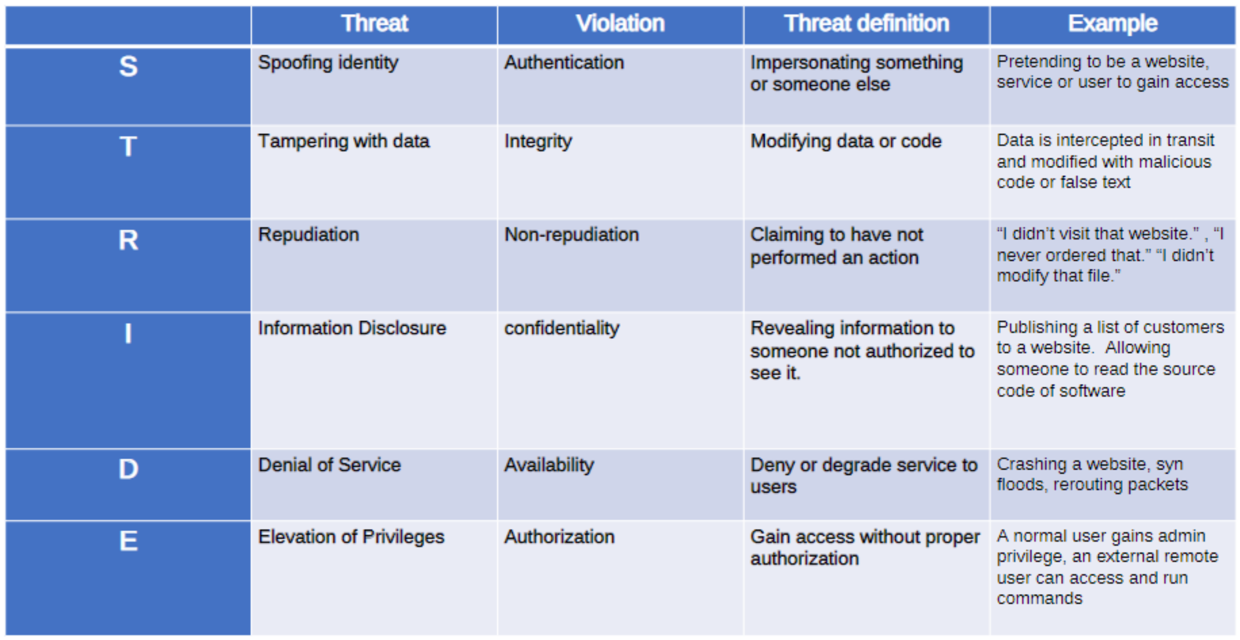
A threat profile is a type of modeling process that illustrates potential threats and attackers that pose a risk to a company's or business's most critical assets. When creating threat profiles, it is important to think like an attacker or someone wanting access to those assets. A systematic evaluation of the available open source tools and resources readily at hand for the average user should be considered, as well as points of entry from both physical and networked locations. The threat intelligence information that is gathered is then used to form individual threat profiles for each asset or threat actor. Threat profiles are important measures of the threats that the organization is facing and are becoming standard practice as part of the software development life cycle (SDLC). Threat profiles inform the cyber security personnel of specific management requirements and help to list out the most pertinent associated risks. In the case of CLSs not only do you have data flowing on the network, you have physical control of the lighting systems themselves as well as the potential to access other connected networks, systems and resources.

Hackers are one of the more commonly known and talked about threat actors, with the general goal of extorting money, private information, or creating denial of service from their targets for personal gain. Advanced persistent threats (APTs) are another group of expert hackers who focus their attention on a specific target. APT groups are often nation state, meaning that their government authorizes the group to launch the attack and may fund those efforts as well. Other threat actors to be aware of are script kiddies, low level, often unskilled hackers playing around or poking the system for fun or sport to see what they can do or access. Hackers and script kiddies however are not the only feasible threats that exist.

Internal threats are also to be considered, such as contractors, employees, engineers etc. A proper evaluation of what the assets are and who might want access to them help to develop a greater understanding and risk profile for the specific objective and resource. Rarely is an attacker able to obtain direct access to the assets they want. It is usually through circumventing or going around security mechanisms already established and in place that access to the desired object or resource is accomplished.

Ideally threat modeling should be used at the earliest stages of development to avoid potential vulnerabilities or costly rebuilds. Once these systems are built, it is equally important to incorporate the NIST cybersecurity frameworks five functions to keep systems resilient from outside influence or tampering. (nist.gov 2018) As technology continues to advance and we see the industry continues to charge ahead by creating cyber physical systems such as smart cities and smart grids, threat models are more important than ever to help mitigate the serious consequences that can come from having the system or network exploited. One of the more well-known models was developed by Microsoft in 1999 and is called STRIDE. The STRIDE model identifies entities, trust boundaries, and assets by creating a data flow diagram that maps out network traffic. This model also helps identify the classes of threats as STRIDE is a mnemonic for Spoofing, Tampering, Repudiation, Information, Denial of service, and Elevation of privileges. (docs.microsoft.com 2017)

Threat profiles are currently being developed for implementation into the overall testing process for the CLSs to aid and facilitate in exploring and characterizing additional vulnerabilities that should be tested.



	Threat	Violation	Threat definition	Example
S	Spoofing identity	Authentication	Impersonating something or someone else	Pretending to be a website, service or user to gain access
T	Tampering with data	Integrity	Modifying data or code	Data is intercepted in transit and modified with malicious code or false text
R	Repudiation	Non-repudiation	Claiming to have not performed an action	“I didn’t visit that website.” , “I never ordered that.” “I didn’t modify that file.”
I	Information Disclosure	confidentiality	Revealing information to someone not authorized to see it.	Publishing a list of customers to a website. Allowing someone to read the source code of software
D	Denial of Service	Availability	Deny or degrade service to users	Crashing a website, syn floods, rerouting packets
E	Elevation of Privileges	Authorization	Gain access without proper authorization	A normal user gains admin privilege, an external remote user can access and run commands

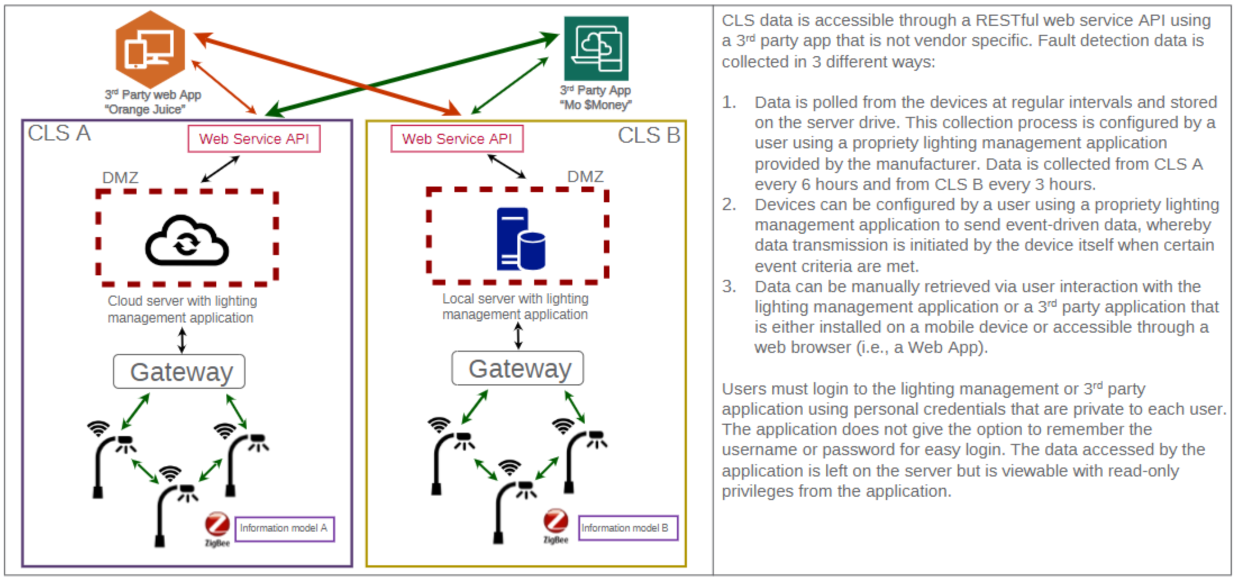
Figure 7. Microsoft Stride Model

The purpose of this model is to continuously define, diagram, Identify, Mitigate and Validate the associated asset and threats against it. The STRIDE model provides a good base for implementing such safeguards. While there are many methods and strategies to develop accurate threat profiles, it is important to pick a method that works effectively for the process and systems that need protection. This could be model specific or a combination of different methodologies to cover a broader or possibly more defined scope of analysis.

10.0 CLS Use Case

Use cases are developed as a visual aid to help analysts understand activities and functions performed by users or devices and simplify in depth modeling processes. Creating a use case is the first step in building a threat profile. For CLSs it was important to create a use case based on a real-world scenario that would show the data flows of lighting systems to better understand where additional vulnerabilities exist outside of authentication and authorization models to incorporate into future tests.

A use case depicting fault detection data accessible through a restful API utilizing third party applications is shown below along with three different ways in which data is collected from CLSs.

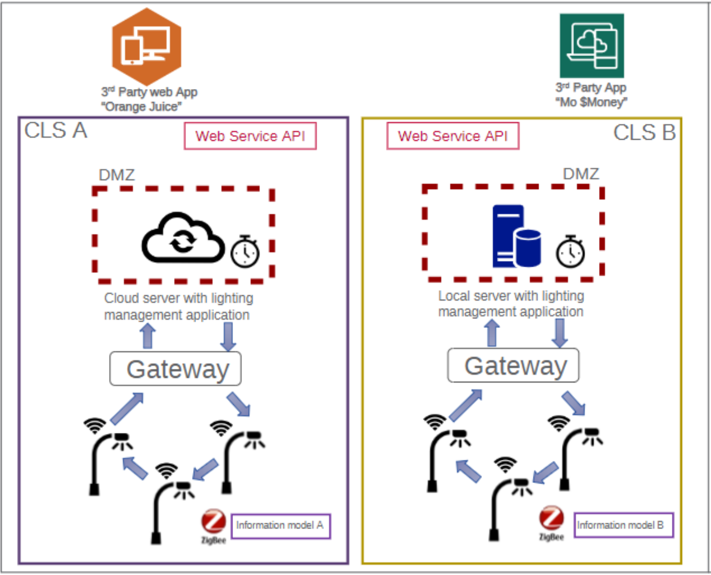

3rd Party web App "Orange Juice"
3rd Party App "Mo $Money"
CLS A
Web Service API
DMZ
Cloud server with lighting management application
Gateway
Information model A
ZigBee
CLS B
Web Service API
DMZ
Local server with lighting management application
Gateway
Information model B
ZigBee

CLS data is accessible through a RESTful web service API using a 3rd party app that is not vendor specific. Fault detection data is collected in 3 different ways:

1. Data is polled from the devices at regular intervals and stored on the server drive. This collection process is configured by a user using a propriety lighting management application provided by the manufacturer. Data is collected from CLS A every 6 hours and from CLS B every 3 hours.
2. Devices can be configured by a user using a propriety lighting management application to send event-driven data, whereby data transmission is initiated by the device itself when certain event criteria are met.
3. Data can be manually retrieved via user interaction with the lighting management application or a 3rd party application that is either installed on a mobile device or accessible through a web browser (i.e., a Web App).

Users must login to the lighting management or 3rd party application using personal credentials that are private to each user. The application does not give the option to remember the username or password for easy login. The data accessed by the application is left on the server but is viewable with read-only privileges from the application.

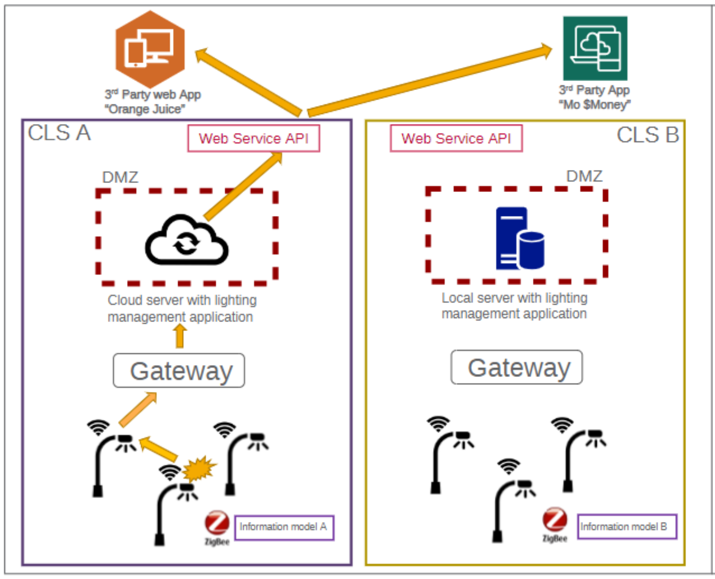
Figure 8. Fault Detection Use Case General Data Flows



The ability of an enterprise software app or tool, provided by a lighting or 3rd party vendor, to access fault detection data (e.g., input voltage out-of-range, below lumen maintenance threshold) from a lighting system, regardless of lighting system vendor, and utilize that data to manage and optimize system maintenance.

Regular interval operation driven by clock

Figure 9. Fault Detection Use Case / Regular Interval Driven by Clock



The ability of an enterprise software app or tool, provided by a lighting or 3rd party vendor, to access fault detection data (e.g., input voltage out-of-range, below lumen maintenance threshold) from a lighting system, regardless of lighting system vendor, and utilize that data to manage and optimize system maintenance.

Event driven data from a device pushed to alert user

Figure 10. Fault Detection Use Case / Event Driven Alert

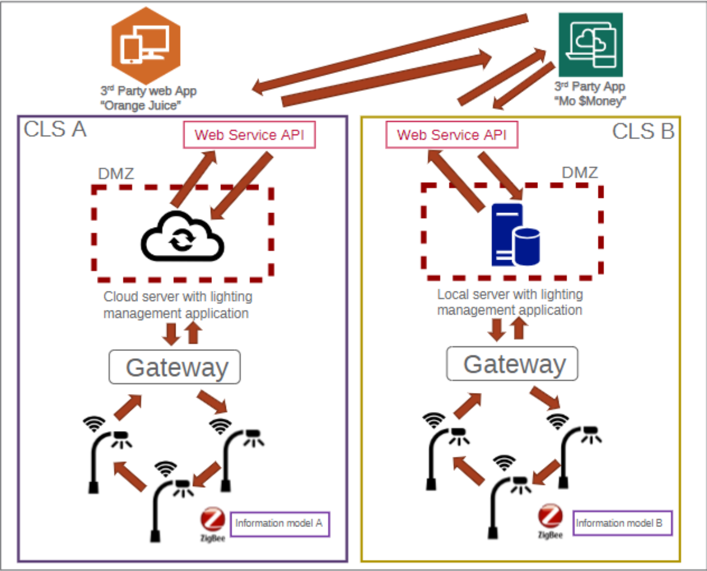
 3rd Party web App "Orange Juice" 3rd Party App "Mo $Money" CLS A Web Service API DMZ Cloud server with lighting management application Gateway ZigBee Information model A CLS B Web Service API DMZ Local server with lighting management application Gateway ZigBee Information model B	The ability of an enterprise software app or tool, provided by a lighting or 3rd party vendor, to access fault detection data (e.g., input voltage out-of-range, below lumen maintenance threshold) from a lighting system, regardless of lighting system vendor, and utilize that data to manage and optimize system maintenance. **Request initiated by user to a single device or system to monitor data**

Figure 11. Fault Detection Use Case / User Initiated Request

11.0 Conclusion

With the newly developed tests and redefined testing guide documentation, in addition to the implementation of use cases to help drive the creation of threat profiles for CLSs, the opportunities to find and document exploits and vulnerabilities that exist within CLSs will help lighting industry stakeholders better understand the challenges they may face when deploying this emerging platform across their networks.

12.0 References

2014. 08 08. Accessed 10 6, 2019. https://www.OWASP.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004).

2019. *CWE.mitre.org.* 06 20. Accessed 11 26, 2019. https://cwe.mitre.org/data/definitions/285.html.

2017. *docs.microsoft.com.* 08 16. Accessed 10 29, 2019. https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats.

n.d. *energy.gov.* Accessed 11 1, 2019. https://www.energy.gov/energysaver/save-electricity-and-fuel/lighting-choices-save-you-money/led-lighting.

2019. *lighting.philips.com.* Accessed 10 28, 2019. https://www.lighting.philips.com/main/inspiration/connected-lighting.

2018. *nist.gov.* 08 10. Accessed 11 12, 2019. https://www.nist.gov/cyberframework/online-learning/five-functions.

2016. *OWASP.org.* 04 1. Accessed 10 1, 2019. https://www.OWASP.org/index.php/Testing_Directory_traversal/file_include_(OTG-AUTHZ-001).

2014. *OWASP.org.* 08 8. Accessed 10 4, 2019. https://www.OWASP.org/index.php/Testing_for_Bypassing_Authorization_Schema_(OTG-AUTHZ-002).

2017. *OWASP.org.* 2 7. Accessed 10 4, 2019. https://www.OWASP.org/index.php/Testing_for_Privilege_escalation_(OTG-AUTHZ-003).

2014. *OWASP.org.* 08 08. Accessed 10 6, 2019. https://www.OWASP.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004).

Purcell, Aaron. 2018. *ibm.com.* 01 16. Accessed 11 9, 2019. https://www.ibm.com/blogs/cloud-computing/2018/01/16/drive-compliance-cloud/.

2019. *Theclimategroup.org.* Accessed 11 26, 2019. https://www.theclimategroup.org/project/led-scale.