Protecting Charging Stations Against Specific Threats

CSP008_S_E

PRESENTATION BY:
DR. ELIAS ATHANASOPOULOS
UNIVERSITY OF CYPRUS
DR. ABDELKADER SHAABAN
AIT AUSTRIAN INSTITUTE OF TECHNOLOGY

Acknowledgement

- *Co-Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or HADEA. Neither the European Union nor the granting authority can be held responsible for them.*
- *Project Agreement no. 101083594*

Protecting Charging Stations Against Specific Threats

Overview

- Topic-1: Introduction to the Energy Charging Infrastructures
- Topic-2: Security Challenges in Energy Charing Stations
- Topic-3: Cascading Effects and impact to other Critical Infrastructures
- **Topic-4: Security Measures and Best Practices for Charging Stations**

AGENDA

1. Best Practices
2. Countermeasures

Best Practices

Charging Stations

They are essentially cyber-physical systems with many components

The components have different architectures that interplay together

- The car may run on several different micro-processors
- The charging station may run on different software/hardware

Components need to establish communication

Layered Security Measures

We can divide a charging station in different layers

- E.g., the system part of the car is on a different layer with the interface that connects to the station

We can then assume different **threat models** per layer

- Each threat model targets different parts of the cyber-physical system

We finally derive security defences for each threat model

- Security defences are not new, but adapted to this setup

Charging Station Layers

System

Realized by low-level code (C/C++) and runs on the core of components

Application

Realized on high-level code (e.g., Web application/service)

Communication

Covers communication between subsystems

System Layer

System code is used to run the low operations of a component

- E.g., the car has multiple microprocessors and on top of that there are operating systems

System code is usually realized in unsafe programming languages (C/C++)

Memory is handled by the developer

The key problem in system code is memory corruption

Memory Corruption

A memory-corruption vulnerability arises when a developer handles memory carelessly

- For example, a memory copy places more data on the destination buffer, than it fits, and the excessive data **corrupt memory**
- Corruption means that the memory will be written with new data that is usually controlled by the attacker

System code, no matter the application, can suffer from memory-corruption vulnerabilities

Memory Corruption in the Charging Station

Memory-corruption is exploited using malicious inputs

In the charging station, there are multiple attackers that can craft malicious inputs

- If the car is vulnerable, then the station may attack the car
- If the station is vulnerable, then the car may attack the station

In all, cases the existing defences are similar

Defending Memory Corruption

Defending memory corruption is an open problem

However, existing hardening techniques exist that can make exploitation harder

Hardening is based on memory-corruption mechanics and attempts to disturb parts of it

Hardening assumes a vulnerability exists and the goal is to prevent exploitation of it

Existing Defenses

Most platforms, operating systems and architectures, support some standard defences

- Non-executable pages
- Stack canaries
- Randomization

Additionally, there are some research proposals for Advanced Hardening

- Some of them (SafeStack, CFI) can be found as options in modern compilers (e.g., in clang)

Some CPU architectures have scheduled defences at the h/w layer

- Intel incorporates shadow stacks in some of their recent CPUs

Non-executable pages

Memory corruption can be used to inject code in a process

- An attacker can stuff code in a malicious input, and if control data is corrupted, then the attacker's code can be executed

Code injection is primarily based on executing data (embedded in malicious inputs)

- For countering code injection, memory pages can be executable but not writable (code pages), or writable but not executable (stack, heap, and other data pages)
- Unfortunately, attackers can use Return-oriented Programming (ROP) and code reuse

Stack canaries

Memory corruption can be easily used when overflowing buffers in the stack of a process

- The common way is to overflow a buffer and change the value of the return address

Stack canaries are random values placed in the stack, between the buffers and the return address

- Changing the return address through a linear overflow will change the value of the canary
- The original value of the canary is stored in a h/w register

Stack canaries can be bypassed using information leaks and vulnerabilities in the heap

Randomization

Code injections can be prevented using non-executable pages; however, exploitation is still possible using ROP

- ROP uses existing code of the process image that assembles as a series of ROP gadgets
- ROP is based on the exact knowledge of the layout of the code
- By randomizing the address space, attackers do not know where ROP gadgets are
- Information leaks can render randomization ineffective

Advanced Hardening

Several techniques have been proposed (e.g., CFI) for making exploitation of memory corruption harder

- Control-flow Integrity computes statically the control-flow graph of a program and then tries to enforce it at run-time
- The CFG contains all legitimate control transfers of the program

CFI can be found in all recent compilers as an option

Memory Corruption - Challenges for Charging Stations

Charing stations include many embedded systems

- It is a cyber-physical system that contains many “small computers” (components)
- Usually, embedded systems run systems code realized in C/C++ and they are vulnerable to memory corruption

Defences are common in more mature systems

- Full-fledged operating systems
- State-of-the art compilers
- Rich in functionality CPUs (e.g., Intel)

Application Layer

System code runs application code, usually developed in higher-level languages

In our setting, Application Code can be a Web service

Charging Stations may offer their functionality as a Web Service

Attacking Web Code

Code Injections (XSS)

- Web applications can suffer from code injections when code is mixed with data
- For a web application, code is expressed in JavaScript that can be stuffed in an HTTP request and be reflected back to another victim's browser

Cross-site Request Forgery (CSRF)

- Malicious web sites can coerce a browser to perform in requests in other web sites where the user maintains an account

Defending Cross-Site Scripting (XSS)

Code Injections can be mitigated by filtering code from data

- This is a hard problem

Another solution is to restrict the sites that are used for loading JavaScript code

- CSP proposed by Mozilla

Defending Cross-site Request Forgery

CSRF can be mitigated using CSRF random tokens

- Sensitive forms can include a random token that should be submitted with the form's action
- The site that needs to protect their users, needs to build more state for generating server-side code (i.e., match random tokens with form submissions)

Other headers have been proposed (Origin header)

Web Attacks – Challenges for Charging Stations

Compared to traditional web sites Charging Stations are closed environments

- They may incorporate web applications, but these can be treated in isolation, compared to the rest of the web ecosystem
- Consider single-site web apps

Countering XSS, CSRF could be considered easier in this setting

- However, there should be caution (e.g., malicious Charging Stations)

Communication Layer

In a Charging Station there is communication involved

Protecting communications from man-in-the-middle attackers can be achieved using encryption

- The standard is TLS
- Beyond standard encryption, TLS needs PKI

Transport Layer Security (TLS)

TLS is the de facto protocol for securing communications using cryptography

- Offers protection for passive and active MitM attackers

Based on symmetric cryptography, asymmetric crryptography and cryptographic hash functions/MACs

Deployed by many applications (web, e-mail, etc.)

Several attacks for TLS, but it is the best we have so far

Public Key Infrastructure (PKI)

TLS, for encrypting communications, needs a PKI infrastructure

- This is an ecosystem for building trust between remote parties

During TLS the two communicating parties need to conclude to a symmetric key

- This key establishment is realized using public-key cryptography
- Trusting a remote public key is done through certificates

Complex ecosystem, with many problems, especially for maintenance

Communications – Challenge for Charging Stations

MitM the communication with a Charging Station is possible

- However, again the system can be considered *closed* compared to the entire Internet

On the other hand, maintain certificates for such closed systems may be considered much harder

Summary

System Layer

Non-executable Data
Randomization
Advanced Hardening

Application Layer

CSP
CSRF Tokens

Communication Layer

TLS

Countermeasures

Priority Recommendations for Potential threats in OCPP-v2.0.1

Overview of recommended mitigations within the analyzed CSMS.

Threats	Mitigation
DoS to CS	• TLSv1.3 under OPCC security profile 3 or IPsec to avoid MiTM actions • Hash functions to verify the integrity of SW components in CSs • Digital signature for each OCPP transaction or the use of Message Authentication Message (MAC) functions • Periodic update of the list of users authorized to change energy in CSs • Redundant mechanisms (e.g.: proxies, communication links) to prevent on-path attacks or authentication in offline mode • Continuous maintenance and certification of HW/SW components • Reputation mechanisms to estimate anomalous user and device behaviors • Data traceability to detect occasional or frequent deviations in one or more transactions • Surveillance and tamper-resistant constructions • Diagnostics, detection, and dynamic event management systems
Manipulation of OCPP CVs	• TLSv1.3 under OPCC security profile 3 or IPSec to avoid MitM actions • Encryption of sensitive data (e.g., OCPP CVs, IDs) • Hash functions to verify the integrity of SW components and data in CSs • Digital signature for each OCPP transaction, or the use of MAC functions • Data traceability to detect occasional or frequent deviations in one or more transactions • Surveillance and tamper-resistant constructions

Alcaraz et al. "OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0." International Journal of Information Security (2023) 22:1395–1421, May 2023.: link OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0 (springer.com)

Priority Recommendations for Potential threats in OCPP-v2.0.1

Overview of recommended mitigations within the analyzed CSMS.

Threats	Mitigation
CS Spoofing	• Diagnostics, detection and dynamic event management systems • TLSv1.3 under OPCC security profile 3 to avoid using identification CVs (Identity and BasicAuthPassword) and MitM actions, or IPSec • Management of unique IDs for each user, device and transaction • Digital signature for each OCPP transaction, or the use of MAC functions • Surveillance and tamper-resistant Constructions • Diagnostics, detection and dynamic event management systems
Manipulation of DERs and storage systems	• Periodically validate power components to verify compliance with regulatory frameworks and international constraints on voltage levels and operational frequency • Continuous maintenance and certification of energy components • Data traceability to detect occasional or frequent deviations in energy components and transactions • Surveillance and tamper-resistant constructions • Diagnostics, detection and dynamic event • management systems

Alcaraz et al. "OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0." International Journal of Information Security (2023) 22:1395–1421, May 2023.: link OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0 (springer.com)

Priority Recommendations for Potential threats in OCPP-v2.0.1

Overview of recommended mitigations within the analyzed CSMS.

Threats	Mitigation
User, CSMS and EMS spoofing	• TLSv1.3 under OPCC security profile 3 to authenticate each part (CSMS, EMS, CS) or IPSec • Management of unique IDs for each user, device and transaction • Awareness to end users about the importance of protecting their credentials and security IDs • Awareness to human operators about the importance of protecting the ecosystem • Access control to the CS under the principles of least privilege and avoid escalation of privilege • Digital signature for each OCPP transaction, or the use of MAC functions • Diagnostics, detection and dynamic event management systems
Information disclosure	• TLSv1.3 under OPCC security profile 3 to authenticate each part (CSMS, CS) or IPSec • Encryption of sensitive data (e.g., OCPP CVs, IDs) • Least privilege principles and segmentation of functions to avoid escalation of privilege • Privacy-enhancing technologies and approaches for the CSMS and EMS

Alcaraz et al. "OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0." International Journal of Information Security (2023) 22:1395–1421, May 2023.: link OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0 (springer.com)

Priority Recommendations for Potential threats in OCPP-v2.0.1

Overview of recommended mitigations within the analyzed CSMS.

Threats	Mitigation
User authorization and admin in CSMS	• TLSv1.3 under OPCC security profile 3 to authenticate each part (CSMS, CS) or IPSec • Valid local authorization lists composed of unique IDs associated with legitimate entities and received from a valid CSMS • Avoid as much as possible the authentication in offline mode and deploy proxies that manage access control via the CSMS • Least privilege principles and segmentation of functions to avoid escalation of privilege • Digital signature for each OCPP transaction, or the use of MAC functions • Diagnostics, detection and dynamic event management systems

Alcaraz et al. "OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0." International Journal of Information Security (2023) 22:1395–1421, May 2023.: link OCPP in the spotlight: threats and countermeasures for electric vehicle charging infrastructures 4.0 (springer.com)

Mitigating Countermeasures

Overview of recommended mitigations within the analyzed CSMS.

CWE-ID	Vulnerability	Mitigation
79	XSS	Sanitize user-controllable input data
89	SQLi	Utilize parameterized queries
200	Information Disclosure	Enforce authentication on all endpoints
306	Missing Auth.	Enforce authentication on all functionalities
321	Embedded Secrets	Request cryptographic keys during runtime
352	CSRF	Utilize random tokens with all requests
425	Forced Browsing	Enforce better access control mechanisms
798	Hard-Coded Cred	Enforce credential update policy
799	Missing Rate Limit	Prevent excessive and fast requests
918	SSRF	Sanitize IP/URL addresses on parameters
942	Cross-Origin Resource Sharing (CORS) Misconfiguration	Enforce stricter cross-domain policy
942	FCDP Misconfiguration	Enforce stricter cross-domain policy
1236	CSVi	Implement safe parsing for CSV files

Nasr, Tony, et al. "Power jacking your station: In-depth security analysis of electric vehicle charging station management systems." *Computers & Security* 112 (2022): 102511.

Connect with CyberSecPro:
How to register and other practical information

1. Website:
   www.cybersecpro-project.eu
2. X (Twitter):
   https://twitter.com/CyberSecPro_eu
3. LinkedIn:
   https://www.linkedin.com/company/cybersecpro-euproject/

Co-funded by the European Union

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or HADEA. Neither the European Union nor the granting authority can be held responsible for them.

Project Agreement no. 101083594

Thank you

Please send all questions to:
Dr. Elias Athanasopoulos
athanasopoulos.elias@ucy.ac.cY

Dr. Abdelkader Shaaban,
abdelkader.Shaaban@ait.ac.at