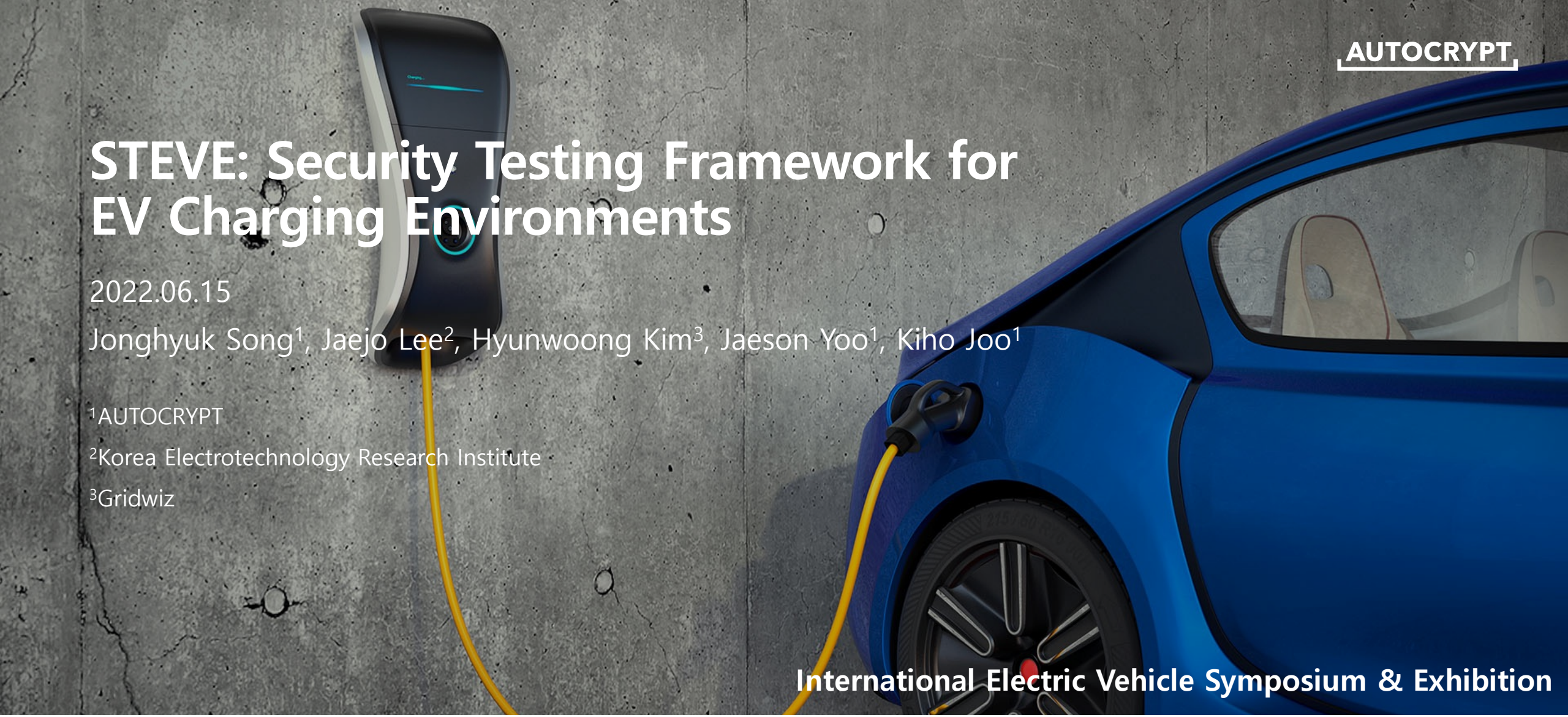


AUTOCRYPT

STEVE: Security Testing Framework for EV Charging Environments

2022.06.15

Jonghyuk Song[1], Jaejo Lee[2], Hyunwoong Kim[3], Jaeson Yoo[1], Kiho Joo[1]

[1]AUTOCRYPT

[2]Korea Electrotechnology Research Institute

[3]Gridwiz

International Electric Vehicle Symposium & Exhibition

TU-Automotive AWARDS
Best Auto Cybersecurity Product/Service 2019
The Automotive Tech Company of the Year Finalist 2020
Automotive Cybersecurity Product of the Year Finalist 2021

CYBER MOBILITY AWARDS
2020 Global Cyber Achievement Award

CTOTECH BREAKTHROUGH AWARDS
2020 & 2021 Automotive Cybersecurity Company of the Year

AIIA
Artificial Intelligence Industry Association
2020 Emerging AI+X Top 100 Company
Mobility Category

Forbes Asia
2021 100 to Watch

ISO 15118: Road Vehicles – Vehicle-to-Grid Communication Interface

Specification about the communication between,

Electric Vehicle(EV) ←→ Electric Vehicle Supply Equipment (EVSE)

Electric Vehicle Communication Controller (EVCC) ←→ Supply Equipment Communication Controller (SECC)

- ISO 15118-1
  - General information and use-case definition
- **ISO 15118-2**
  - **Network and application protocol requirements**
- ISO 15118-3
  - Physical and data link layer requirements

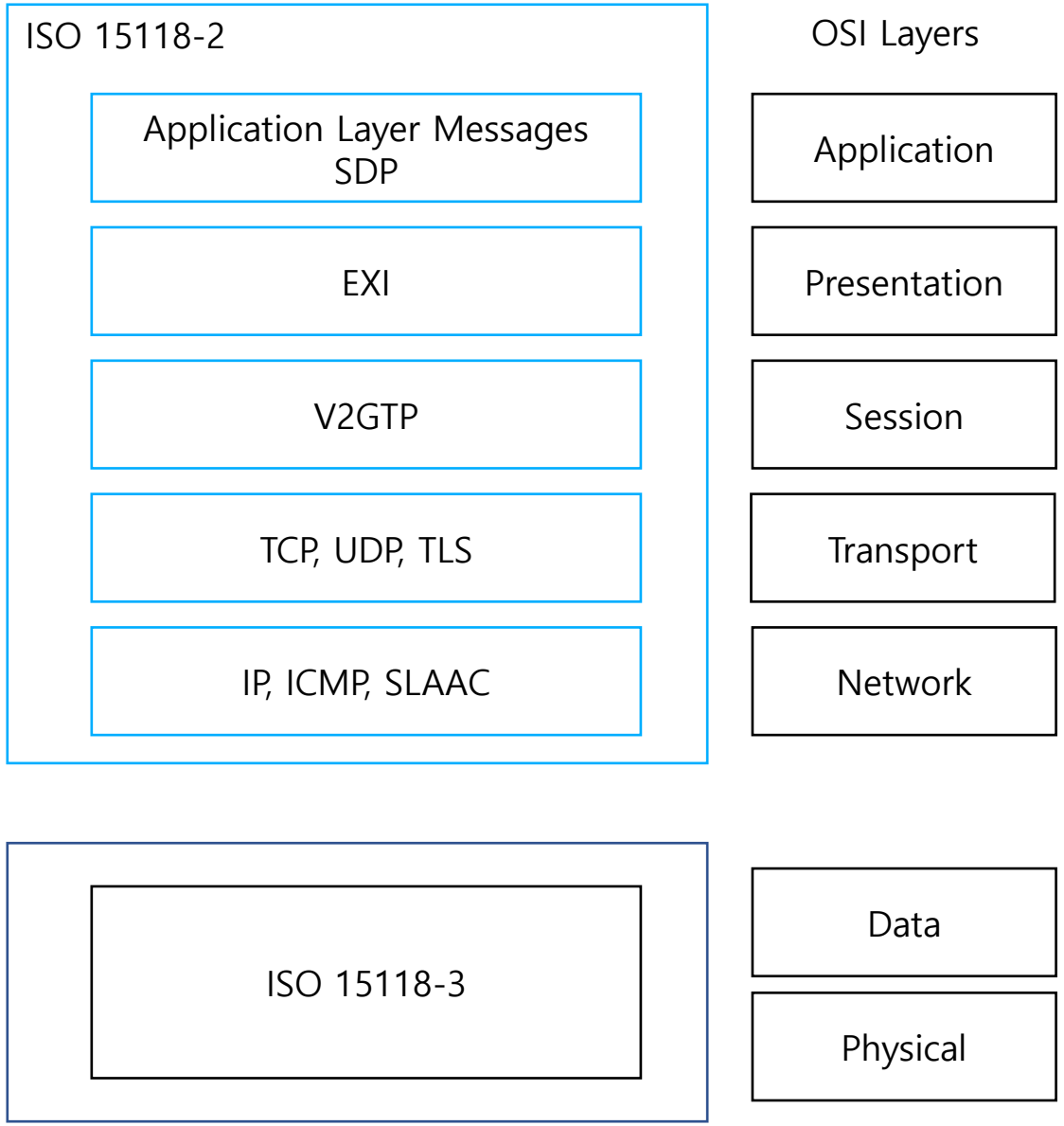
- ISO 15118-4 (under preparation)
  - Network and application protocol conformance test
- ISO 15118-5 (under preparation)
  - Physical layer and data link layer conformance test

V2G Communication Overview

SDP → SECC Discovery Protocol
An EVCC uses the SDP to get the IP, port of the SECC.

EXI → Efficient XML Interchange
Encode XML in a binary format

V2GTP → V2G Transfer Protocol
Transfer protocol between the EVCC and SECC



Cybersecurity Attacks in ISO 15118-2

- Eavesdropping
  - Private data could be leaked
- Manipulation of charging data
  - Charging more than you paid
- Masquerading
  - Pass the fee to another user

Type	Name
Identification	EVCCID, EVSEID, SessionID
Charge	MeterInfo, EVMaxVoltage
Tariff	SalesTariffID

< Examples of sensitive parameters in ISO 15118-2>

Cybersecurity Attacks in ISO 15118-2

- Denial of Service Attack
  - Traditional network DOS attacks could be possible.
    e.g., SYN flooding

- XML Injection Attack
  - Traditional XML injection attacks may be possible.
    e.g., XXE injection attack

- Invalid Data Attack
  - Malformed XMLs could crash the parser.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns6:V2G_Message xmlns:ns6="urn:iso:15118:2:2013:MsgDef"
xmlns:ns5="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns7="urn:iso:15118:2:2013:MsgBody"
xmlns:ns2="urn:iso:15118:2:2010:AppProtocol"
xmlns:ns4="urn:iso:15118:2:2013:MsgDataTypes"
xmlns:ns3="urn:iso:15118:2:2013:MsgHeader">
<ns6:Header>
    <ns3:SessionID>3CEFDB85233A321EF</ns3:SessionID>
</ns6:Header>
<ns6:Body>
    <ns7:ChargingStatusRes>
        <ns7:ResponseCode>OK</ns7:ResponseCode>
        <ns7:EVSEID>KR*V2G*K12345</ns7:EVSEID>
        <ns7:SAScheduleTupleID>1</ns7:SAScheduleTupleID>
        <ns7:MeterInfo>
            <ns4:MeterID>1</ns4:MeterID>
            <ns4:MeterReading>32000</ns4:MeterReading>
            <ns4:TMeter>1654232745</ns4:TMeter>
        </ns7:MeterInfo>
        <ns7:ReceiptRequired>false</ns7:ReceiptRequired>
            <ns7:AC_EVSEStatus>
                <ns4:NotificationMaxDelay>0</ns4:NotificationMaxDelay>
                <ns4:EVSENotification>None</ns4:EVSENotification>
                <ns4:RCD>false</ns4:RCD>
            </ns7:AC_EVSEStatus>
    </ns7:ChargingStatusRes>
</ns6:Body>
</ns6:V2G_Message>
```

<XML message example in ISO 15118-2>

How to prevent the attacks??

SECC/EVCC should implement countermeasures for each attack.

We need to verify that the countermeasures are implemented correctly.

How??

Countermeasures

1. Correct use of TLS
2. Validation on charging parameters
3. Validation on identification parameters
4. DoS attacks prevention
5. XML injection prevention
6. Data validation

STEVE: Security Testing Framework for EV Charging Environments

STEVE checks whether the all countermeasures are correctly implemented on SECC/EVCC.

How?

Run the real attacks

Check whether the attacks are successful or not

- Contributions
  - Define potential cybersecurity threats in ISO 15118
  - Propose the countermeasures for each attack
  - Provide methods to check whether SECC/EVCC can prevent the attacks

Countermeasure 1. Correct Use of TLS

- It is most important to make sure that TLS is being used properly.

- STEVE checks,
  1. TLS is available? → SECC/EVCC should support TLS.

  2. Latest TLS version is being used? → TLS v1.2/v1.3 should be used.

  3. Certificate is not expired? → Expired certificate should not be used.

OpenSSL commands help us to check TLS

```
jhsong@jhsong-macbook:~$ openssl s_client -connect fe80:0:0:0:485:2a30:b42e:5d20%en0:64872
CONNECTED(00000003)
depth=2 CN = CPOSubCA1, O = RISE V2G Project, C = DE, DC = V2G
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/CN=SECCCert/O=RISE V2G Project/C=DE/DC=CPO
   i:/CN=CPOSubCA2/O=RISE V2G Project/C=DE/DC=V2G
 1 s:/CN=CPOSubCA2/O=RISE V2G Project/C=DE/DC=V2G
   i:/CN=CPOSubCA1/O=RISE V2G Project/C=DE/DC=V2G
 2 s:/CN=CPOSubCA1/O=RISE V2G Project/C=DE/DC=V2G
   i:/CN=V2GRootCA/O=RISE V2G Project/C=DE/DC=V2G
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIBzjCCAXWgAwIBAgICMDkwCQYHKoZIzj0EATBRMRIwEAYDVQQDDAlDUE9TdWJD
QTIxGTAXBgNVBAoMEFJJU0UgVjJHIFByb2plY3QxCzAJBgNVBAYTAkRFMRMwEQYK
CZImiZPyLGQBGRYDVjJHMB4XDTIyMDYwMzA0Mzg1OFoXDTIyMDgwMjA0Mzg1OFow
UDERMA8GA1UEAwwIU0VDQ0NlcnQxGTAXBgNVBAoMEFJJU0UgVjJHIFByb2plY3Qx
CzAJBgNVBAYTAkRFMRMwEQYKCZImiZPyLGQBGRYDQ1BPMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAE8mvUtuxAigGGdcP2mQYBEE45mhc8nCdmOfsWCRr1xsLlJ6rj
LIu/vZLXJR84eMhhuNwy5Wo0gRZfdT/3LzP6c6M/MD0wDAYDVR0TAQH/BAIwADAO
BgNVHQ8BAf8EBAMCA4gwHQYDVR0OBBYEFOvH2xrR3Yx0G4Z3whksJj40HGTQMAkG
ByqGSM49BAEDSAAwRQIgP2JWvR/4ZYmcfIR41y33LP834D2x8Sdxms2pWffGWzwC
IQDHOZk8YySvjN7x++s7iubEr36H2j/jbOyDm1YpUmQB3g==
-----END CERTIFICATE-----
subject=/CN=SECCCert/O=RISE V2G Project/C=DE/DC=CPO
issuer=/CN=CPOSubCA2/O=RISE V2G Project/C=DE/DC=V2G
---
No client certificate CA names sent
Server Temp Key: ECDH, X25519, 253 bits
---
SSL handshake has read 1738 bytes and written 329 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-ECDSA-AES128-SHA256
Server public key is 256 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-ECDSA-AES128-SHA256
    Session-ID: C814616477A434CA4CD6BCEB495066FD78EA5C152866EDE47ADC5F291EC1FAF8
    Session-ID-ctx:
    Master-Key: CA48BF83A3A40801FCE2C00FF52A2FB9B18A5F503AFEC998F2C62D35F23B06C26C7537EDEB92AA
    Start Time: 1654677366
    Timeout   : 7200 (sec)
    Verify return code: 20 (unable to get local issuer certificate)
```

Certificate

TLS version info

```
$ openssl s_client -connect [IP]:[port]
```

```
jhsong@jhsong-macbook:~$ openssl s_client -connect fe80:0:0:0:485:2a30:b42e:5d20%en0:52979
CONNECTED(00000003)
```

If TLS is unavailable, TLS/certificate information is not printed.

OpenSSL commands help us to check TLS

```
jhsong@jhsong-macbook:~$ openssl x509 -enddate -noout -in cert.pem
notAfter=Aug  2 04:38:58 2022 GMT
```

Expiration date

- OpenSSL can check certificate expiration date.

Countermeasure 2. Validation on Charging Parameters

SECC/EVCC should validate charging parameters.

To check the validation,

STEVE transmits the parameters that violates the types, or the restrictions defined in ISO 15118-2.

Parameter Name	Type (restriction)
VersionNumberMajor	unsignedInt
SchemaID	unsignedByte
ServiceScope	String (max length: 64)
SelectedPaymentOption	Enumeration (Contract, ExternalPayment)
RequestedEnergyTransferMode	Enumeration (AC_single_phase_core, AC_three_phase_core, DC_core, DC_combo_core, …)
ChargeProgress	Enumeration (Start, Stop, Renegotiate)

< Example of charging parameters in ISO 15118-2 >

Countermeasure 3. Validation on Identification Parameters

Identification parameters also should be validated.

Parameter Name	Type (restriction)
SessionID	hexBinary (length <=8)
EVCCID	hexBinary (length <=6)
EVSEID	String (3 <= length<= 7)
ServiceID	unsignedShort

< Example of identification parameters in ISO 15118-2 >

Malicious Values

Sample malicious values transmitted by STEVE

→ (null), “A”*1024, 0, -1, 4294967296 (unsigned int max+1), 18,446,744,073,709,551,616 (unsigned long max+1), …

```
<ns0:V2G_Message xmlns:ns0="urn:iso:15118:2:2013:MsgDef"
xmlns:ns1="urn:iso:15118:2:2013:MsgHeader" xmlns:ns2="urn:iso:15118:2:2013:MsgBody">
<ns0:Header>
<ns1:SessionID>
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA
</ns1:SessionID>
</ns0:Header>
<ns0:Body>
<ns2:SessionSetupReq>
<ns2:EVCCID>A07817A65779</ns2:EVCCID>
</ns2:SessionSetupReq>
</ns0:Body>
</ns0:V2G_Message>
```

* Example of normal SessionID: C3917A55A2BA8118

```
<ns0:V2G_Message xmlns:ns0="urn:iso:15118:2:2013:MsgDef" xmlns:ns1="urn:iso:15118:2:2013:MsgHeader"
xmlns:ns2="urn:iso:15118:2:2013:MsgBody" xmlns:ns3="urn:iso:15118:2:2013:MsgDataTypes">
<ns0:Header>
<ns1:SessionID>74F18B6F1F222CA5</ns1:SessionID>
</ns0:Header>
<ns0:Body>
<ns2:PowerDeliveryReq>
<ns2:ChargeProgress>-1</ns2:ChargeProgress>
<ns2:SAScheduleTupleID>1</ns2:SAScheduleTupleID>
<ns2:ChargingProfile>
<ns3:ProfileEntry>
<ns3:ChargingProfileEntryStart>0</ns3:ChargingProfileEntryStart>
<ns3:ChargingProfileEntryMaxPower>
<ns3:Multiplier>3</ns3:Multiplier>
<ns3:Unit>W</ns3:Unit>
<ns3:Value>11</ns3:Value>
</ns3:ChargingProfileEntryMaxPower>
<ns3:ChargingProfileEntryMaxNumberOfPhasesInUse>3</ns3:ChargingProfileEntryMaxNumberOfPhasesInUse>
</ns3:ProfileEntry>
</ns2:ChargingProfile>
</ns2:PowerDeliveryReq>
</ns0:Body>
</ns0:V2G_Message>
```

* Example of ChargeProgress: “Start” or “Stop”

Countermeasure 4. DoS Attack Prevention

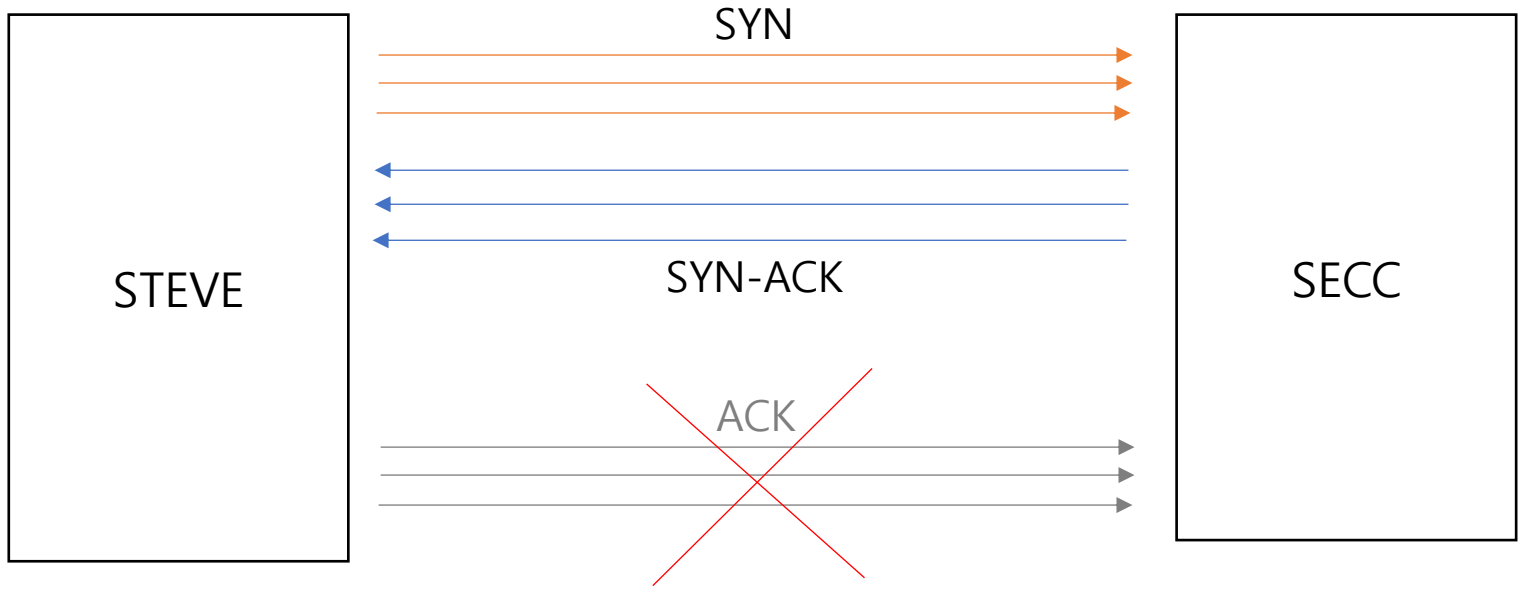
ISO 15118-2 is based on TCP/IP, so traditional DoS could be possible.

STEVE performs flooding attacks to check DoS attack prevention.

We had to convert flooding attack codes to IPv6. (ISO 15118-2 is based on IPv6)

One of the example traditional DoS → SYN flooding attack

Attackers send a large number of "SYN" packets to the server but does not send "ACK" back to the server.



Countermeasure 5. XML Injection Prevention

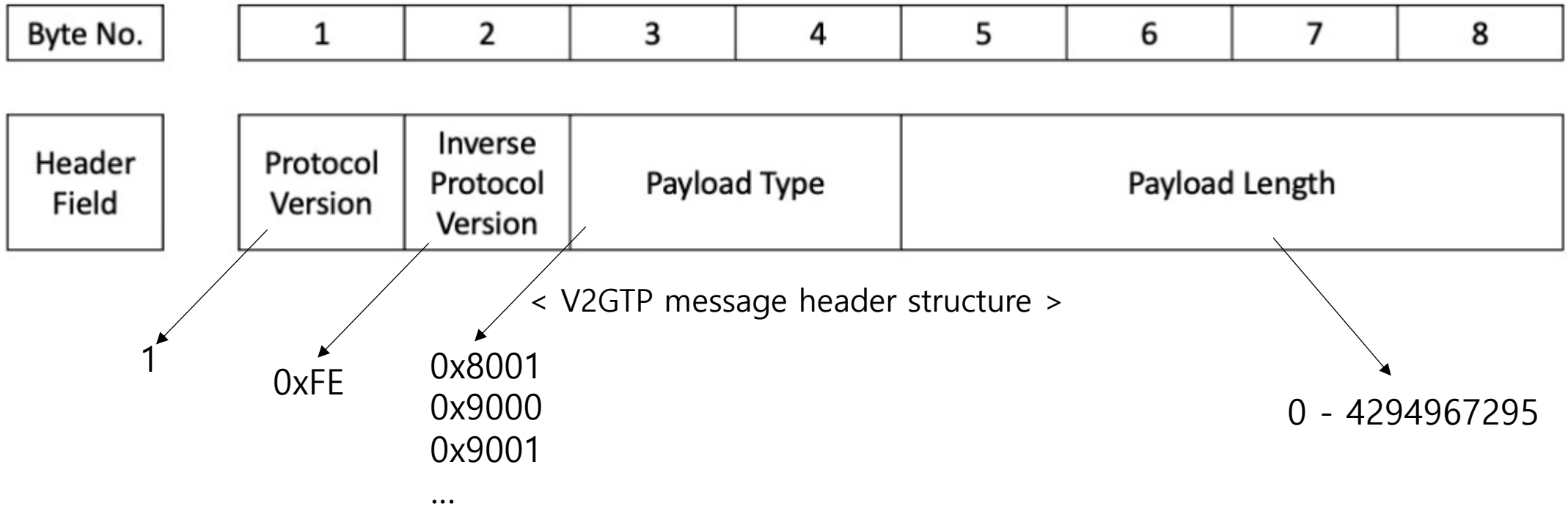
STEVE transmits malicious XML data to check that XML injection is possible.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

< Example of XXE injection attack payload>

Countermeasure 6. Data Validation

STEVE transmits invalid V2GTP messages to check whether the SECC/EVCC handle exceptions properly.



Test Procedures

1. Preparation process

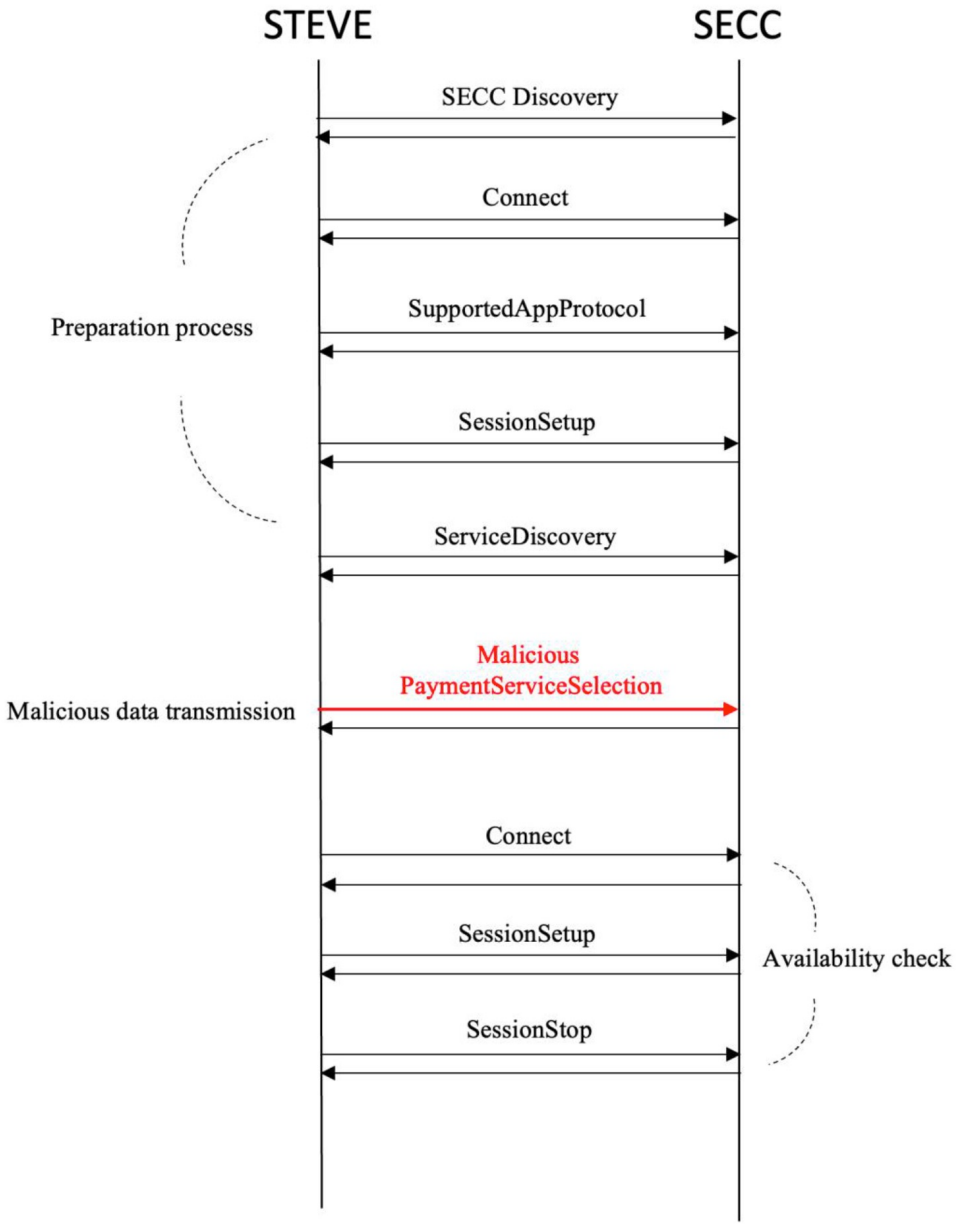
Proceed normal communication

2. Malicious data transmission

Send malicious data

3. Check

Check the availability of the target



Results

Implementation

STEVE is implemented in Python3 and Java.

Experiments

STEVE found several exceptions in RISE-V2G.

After the exceptions occurred, RISE-V2G-SECC cannot accept the normal connection from normal EVCC.

Exceptions	Description
java.lang.IllegalArgumentException	hexBinary needs to be even-length: 0
	contains illegal character for hexBinary: -1
java.lang.NumberFormatException	Not a Number: <
	For input string: "184467440737095516l7"
java.lang.NullPointerException	-

< Exceptions in RISE-V2G founded by STEVE >

Exception examples

```
2022-06-10T13:47:18,836 DEBUG [ConnectionThread fe80:0:0:0:485:2a30:b42e:5d20%14] ConnectionHandler: Message received
2022-06-10T13:47:18,836 DEBUG [ConnectionThread fe80:0:0:0:485:2a30:b42e:5d20%14] EXIficientCodec: Received EXI stream: 8098004011D70216988000
2022-06-10T13:47:18,837 ERROR [ConnectionThread fe80:0:0:0:485:2a30:b42e:5d20%14] EXIficientCodec: RuntimeException occurred while trying to unmarshall
java.lang.RuntimeException: java.lang.IllegalArgumentException: contains illegal character for hexBinary: -1
        at com.v2gclarity.risev2g.shared.messageHandling.MessageHandler$1.handleEvent(MessageHandler.java:321) ~[rise-v2g-secc-1.2.6.jar:?]
        at com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext.handleEvent(UnmarshallingContext.java:702) ~[rise-v2g-secc-1.2.6.jar:?]
        at com.sun.xml.bind.v2.runtime.unmarshaller.Loader.reportError(Loader.java:232) ~[rise-v2g-secc-1.2.6.jar:?]
        at com.sun.xml.bind.v2.runtime.unmarshaller.Loader.handleGenericException(Loader.java:219) ~[rise-v2g-secc-1.2.6.jar:?]
        at com.sun.xml.bind.v2.runtime.unmarshaller.LeafPropertyLoader.text(LeafPropertyLoader.java:37) ~[rise-v2g-secc-1.2.6.jar:?]
        at com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext.text(UnmarshallingContext.java:558) ~[rise-v2g-secc-1.2.6.jar:?]
        at com.sun.xml.bind.v2.runtime.unmarshaller.SAXConnector.processText(SAXConnector.java:182) ~[rise-v2g-secc-1.2.6.jar:?]
```

```
:iso:15118:2:2010:AppProtocol" xmlns:ns4="urn:iso:15118:2:2013:MsgDataTypes" xmlns:ns3="urn:iso:15118:2:2013:MsgHeader"><ns6:Header><ns3:Sess
liveryReq><ns7:SAScheduleTupleID>1</ns7:SAScheduleTupleID><ns7:ChargingProfile><ns4:ProfileEntry><ns4:ChargingProfileEntryStart>0</ns4:Chargi
ns4:Multiplier><ns4:Unit>W</ns4:Unit><ns4:Value>11</ns4:Value></ns4:ChargingProfileEntryMaxPower><ns4:ChargingProfileEntryMaxNumberOfPhasesIn
</ns7:ChargingProfile></ns7:PowerDeliveryReq></ns6:Body></ns6:V2G_Message>
2022-06-10T13:35:40,149 DEBUG [ConnectionThread fe80:0:0:0:485:2a30:b42e:5d20%14] WaitForPowerDeliveryReq: PowerDeliveryReq received
Exception in thread "ConnectionThread fe80:0:0:0:485:2a30:b42e:5d20%14" java.lang.NullPointerException
        at com.v2gclarity.risev2g.secc.states.WaitForPowerDeliveryReq.isResponseCodeOK(WaitForPowerDeliveryReq.java:115)
        at com.v2gclarity.risev2g.secc.states.WaitForPowerDeliveryReq.processIncomingMessage(WaitForPowerDeliveryReq.java:65)
        at com.v2gclarity.risev2g.secc.session.V2GCommunicationSessionSECC.processIncomingMessage(V2GCommunicationSessionSECC.java:184)
        at com.v2gclarity.risev2g.secc.session.V2GCommunicationSessionSECC.update(V2GCommunicationSessionSECC.java:162)
        at java.base/java.util.Observable.notifyObservers(Observable.java:173)
        at com.v2gclarity.risev2g.secc.transportLayer.ConnectionHandler.run(ConnectionHandler.java:136)
        at java.base/java.lang.Thread.run(Thread.java:829)
2022-06-10T13:35:45,151 DEBUG [UDPServerThread] UDPServer: Message received
2022-06-10T13:35:45,152 DEBUG [UDPServerThread] V2GCommunicationSessionHandlerSECC: SECCDiscoveryReq received
```

Conclusions

SECC/EVCC supporting ISO 15118 could be exposed to several cyber attacks

To checks if SECC/EVCC are vulnerable to the attacks,

STEVE performs the attacks and checks whether SECC/EVCC prevent them.

AUTOCRYPT

Secure First, Then Ride

SEOUL · SEJONG · SHANGHAI · WUXI · TORONTO · TOKYO

www.autocrypt.io