The research on vulnerability analysis in OpenADR for Smart Grid

[1]Mijeong Park, [1]Miyoung Kang, and [1]Jin-Young Choi

[1]Department of Embedded Software, Korea University, Seoul, Korea
{mjpark,mykang,choi}@formal.korea.ac.kr

Abstract. Smart Grid has become more important for the efficient use of electric power and the demand reduction. As demand for electric power is increasing continuously despite its limited capacity. The demand reduction in Smart Grid can be achieved through DR(Demand Response) which reduces demand for electric power. In this paper, we analyzed the weaknesses of open source of Open ADR, protocol for Smart Grid DR, using CERT Java secure coding rules. We extracted the violations of rules such as OBJ01-J that means the scope of declaring member variables which should be obeyed in Object-Oriented Programming and IDS00-J that means the validation for input data which should be obeyed in Web environment. By eliminating the weaknesses we could enhance the security of Smart Grid communications.

Keywords: Smart Grid, Demand Response, OpenADR, Open Source, Vulnerability, Security Weakness, Secure Coding

1 Introduction

Smart Grid is gaining popularity these days since it can solve the problems of increasing demand of electric power, improving the green energy industry at the same time. In addition, the importance of Smart Grid is increasing because of the possibility of blackout caused by the absence of efficient power system management, which can result in economic and national loss.

Smart Grid requires the secure communication environment for the data integrity due to its network-based characteristics. In particular, hacking attacks such as information leakage and manipulation of data can occur, while sending or receiving information between suppliers and consumers. Additionally, Smart Grid is made of combined structure of several systems, with several external connections. As we could see the Stuxnet[1] in 2010 where Iran nuclear power plant got attacked, Smart Grid, a national infrastructure, can be attacked by malicious hackers. It makes the needs of secure system in Smart Grid essential.

In this paper, we suggest several techniques not only to improve security and stability but also to remove systematic weaknesses in Smart Grid by applying secure coding. Also, we propose new secure coding rules for Smart Grid Software, especially when HeartBleed[2] vulnerability of recent OpenSSL Open source has caused a huge amount of loss around the world. It clearly shows that it is necessary to analyze vulnerability of Open source used in Smart Grid. We use CERT Java secure coding rules[3] and LDRA Testbed[4] for static analysis based on CERT Java.

The rest of this paper proceeds as follows: Section 2 explains the related works on Smart Grid, Secure Coding and analysis tools. Section 3 shows the weaknesses derived from secure coding rules in AMI(Advanced Metering Infrastructure) Software(open source of OpenADR[5]). Section 4 explains conclusion and future works.

.

2 Related works

2.1 Secure Coding

CWE(Common Weakness Enumeration)[6] is the standard for measuring software security weaknesses around the world in order to improve security and quality of software. Although it is impossible to eliminate all security weaknesses, we need to put in an effort to minimize those security weaknesses. Secure Coding is the coding standard of source code for developing secure software free of those security weaknesses.

There are a variety of rules in secure coding depending on the characteristics of organizations or fields of study such as Secure Coding Guide developed by the Ministry Of Security and Public Administration in Korea, CERT C/Java developed by the Software Engineering Institute in Carnegie Mellon University and so on.

In this paper, CERT Java is used to analyze open source of OpenADR[8] that is a protocol of Smart Grid, implemented in java.

2.2 Analysis tool

Quality of software can be improved through static and dynamic analysis of the entire software development process. In this paper, we utilize LDRA TestBed tool[7] for the static analysis that tests source code without executing the program. LDRA's proprietary parsing engine allows us to quickly incorporate new analysis techniques to meet changing standards requirements. Therefore, LDRA Testbed enforces compliance with coding standards and clearly indicates software flaws that might otherwise pass through the standard build and test process to become latent problems. We analyzed whether security weaknesses are removed according to CERT Java secure coding rules on LDRA TestBed.

3 Analyzing security weaknesses based on CERT Java

3.1 Analysis target

Providers and receivers can transmit and receive data back and forth through the Internet in Smart Grid. In this environment, the critical data such as electric power information and private information can be transferred. During this process, we need to eliminate security weaknesses of the software in order to prevent malicious attacks from hackers.

The analysis target is open source software of OpenADR facilitating data transfer between provider(VTN) and receiver(VEN) in Smart Grid as shown in Figure 1. There is open source software based on XML for OpenADR.

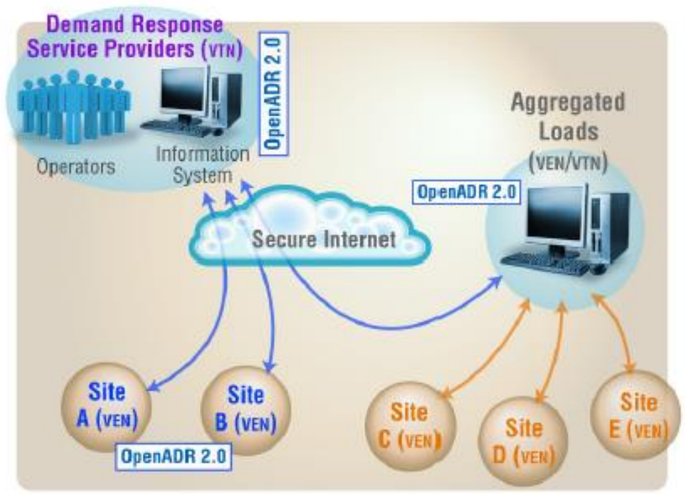


Fig. 1. OpenADR model

We utilize LDRA Testbed 9.4.3 and analyze open source software of OpenADR based on CERT Java secure coding rules. Through this analysis, we detected violations of CERT Java secure coding rules in terms of Static, Complexity and Data Flow.

3.2 Analysis result

The analysis result on open source software of OpenADR based on CERT Java is shown in table 1. 'CERT code' means the rule code of CERT Java secure coding and 'Number of Violations' means the number of violations of those rules. From this result, we can see the violations of secure coding rules.

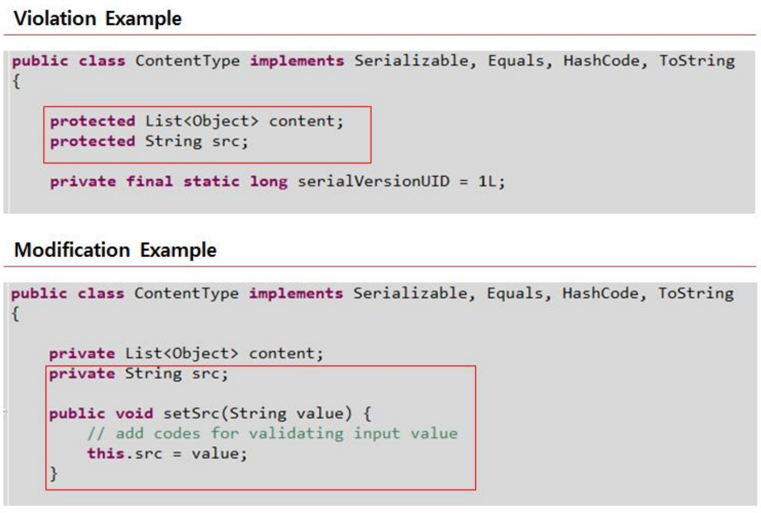
CERT code	Number of Violations (VEN)	Number of Violations (VTN)	Total
IDS05-J	0	0	**0**
IDS06-J	0	0	**0**
EXP00-J	0	2	**2**
EXP05-J	130	5	**135**
EXP06-J	0	0	**0**
NUM00-J	0	0	**0**
NUM01-J	0	0	**0**
NUM02-J	0	0	**0**
NUM07-J	0	0	**0**
NUM09-J	0	0	**0**
NUM12-J	0	5	**5**
NUM13-J	0	0	**0**
OBJ01-J	627	93	**720**
OBJ10-J	404	28	**432**
MET02-J	0	0	**0**
ERR03-J	2	3	**5**
THI00-J	0	0	**0**
THI05-J	0	0	**0**
FIO02-J	0	0	**0**
FIO04-J	0	0	**0**
FIO09-J	0	0	**0**
MSC01-J	7	7	**14**
MSC02-J	0	0	**0**

Table 1. The result about violations of secure coding in open source of OpenADR

'OBJ01-J'[9] rule is to declare data members as private and provide accessible wrapper methods. The data members(variables) should be declared as private to prevent the change of data members in unexpected ways. Open source software of OpenADR violated more than 720 rules. Especially, changing data members in unexpected ways can cause serious problems in OpenADR transferring information about electric power usage between providers and receivers.

'OBJ01-J' rule is also related to 'CWE766'[10], the security weakness declaring critical variable as public. 'CVE-2010-3860'[11] is an actual example where the critical data such as user name and directory path were hacked by a remote attacker since they were declared in public.

Violation Example of Figure 2 is the violated part of 'OBJ01-J' rule in open source of OpenADR. To resolve this security weakness, we need to declare the critical variables as private and provide the public method as shown Modification Example in Figure 2. By applying this method we can prevent direct access to these variables.



Violation Example

```
public class ContentType implements Serializable, Equals, HashCode, ToString
{

    protected List<Object> content;
    protected String src;

    private final static long serialVersionUID = 1L;
```

Modification Example

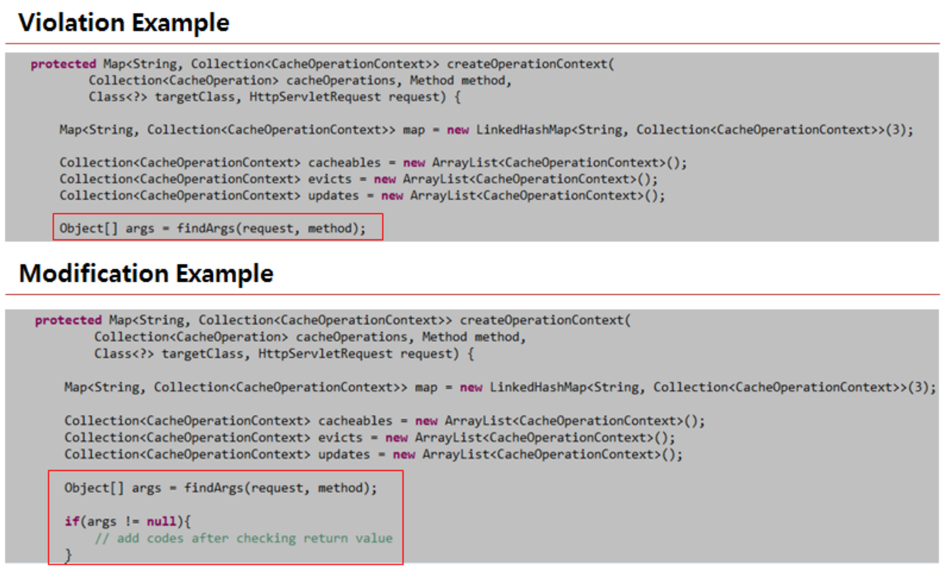
```
public class ContentType implements Serializable, Equals, HashCode, ToString
{

    private List<Object> content;
    private String src;

    public void setSrc(String value) {
        // add codes for validating input value
        this.src = value;
    }
```

Fig. 2. The solution of OBJ01-J rule violation in open source of OpenADR

‘EXP00-J’[12] rule is not to ignore the values returned by methods. The method returns the value to notify success/failure or to update the value. If we ignore or don't handle the returned value from the method properly, it can cause security problems. Therefore, we should handle the return value of the method in a proper manner to prevent the security problems in advance.

'EXP00-J' rule is also related to ‘CWE252’[13], the security weakness caused by not checking the returned value from the method. ‘CVE-2010-0211’[14] is an actual case of DoS(Denial of Service) attack resulting from the violation of 'EXP00-J' rule in OpenLDAR(open source of LDAP).

Violation Example in Figure 3 is the violated part of 'EXP00-J' rule in open source software of OpenADR. To resolve this security weakness, we should handle the return value from the method properly as shown in Modification Example of Figure 3.


Violation Example

```
protected Map<String, Collection<CacheOperationContext>> createOperationContext(
        Collection<CacheOperation> cacheOperations, Method method,
        Class<?> targetClass, HttpServletRequest request) {

    Map<String, Collection<CacheOperationContext>> map = new LinkedHashMap<String, Collection<CacheOperationContext>>(3);

    Collection<CacheOperationContext> cacheables = new ArrayList<CacheOperationContext>();
    Collection<CacheOperationContext> evicts = new ArrayList<CacheOperationContext>();
    Collection<CacheOperationContext> updates = new ArrayList<CacheOperationContext>();

    Object[] args = findArgs(request, method);
```

Modification Example

```
protected Map<String, Collection<CacheOperationContext>> createOperationContext(
        Collection<CacheOperation> cacheOperations, Method method,
        Class<?> targetClass, HttpServletRequest request) {

    Map<String, Collection<CacheOperationContext>> map = new LinkedHashMap<String, Collection<CacheOperationContext>>(3);

    Collection<CacheOperationContext> cacheables = new ArrayList<CacheOperationContext>();
    Collection<CacheOperationContext> evicts = new ArrayList<CacheOperationContext>();
    Collection<CacheOperationContext> updates = new ArrayList<CacheOperationContext>();

    Object[] args = findArgs(request, method);

    if(args != null){
        // add codes after checking return value
    }
```

Fig. 3. The solution of EXP00-J rule violation in open source of OpenADR

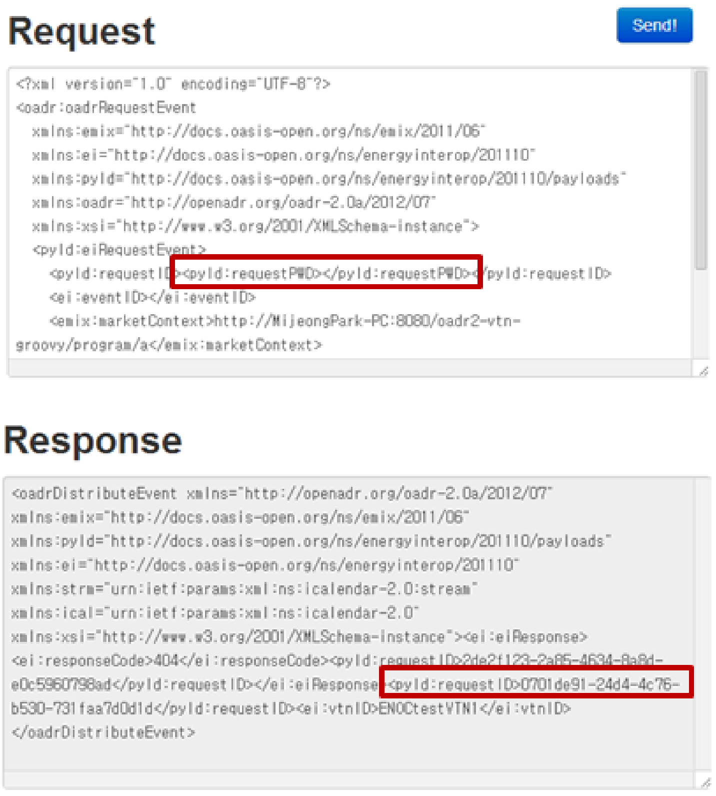
After modifying open source of OpenADR according to CERT Java Secure Coding Rules, we confirmed that the program of open source of OpenADR was executed successfully.

3.3 Analysis result through setting up the test environment

Smart Grid requires the web environment since it transfers data through the common networks between providers and receivers. For this reason, complying with 'Input Validation and Data Sanitization (IDS)' rule of CERT Java secure coding is essential.

The set of IDS rules includes a total of 14 kinds of rules, 'IDS00-J(Sanitize untrusted data passed across a trust boundary)' rule and IDS11-J(Eliminate non-character code points before validation) rule are especially important among them.

We set up the test environment to find the security weaknesses that cannot be detected by the LDRA tool. Figure 4 is the result of testing Request and Response using open source software of OpenADR between providers and receivers. This test is the example of the wrong input value intentionally made. In other words, it enters XML tag as a value of requestID field, resulting in unexpected value as shown in Figure 4 Response. This result can cause huge damage where we are not able to control correct usage of electric power under emergency in Smart Grid.

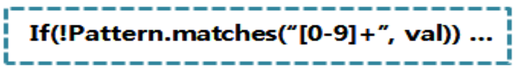


```
<?xml version="1.0" encoding="UTF-8"?>
<oadr:oadrRequestEvent
  xmlns:emix="http://docs.oasis-open.org/ns/emix/2011/06"
  xmlns:ei="http://docs.oasis-open.org/ns/energyinterop/201110"
  xmlns:pyld="http://docs.oasis-open.org/ns/energyinterop/201110/payloads"
  xmlns:oadr="http://openadr.org/oadr-2.0a/2012/07"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <pyld:eiRequestEvent>
    <pyld:requestID><pyld:requestPWD></pyld:requestPWD></pyld:requestID>
    <ei:eventID></ei:eventID>
    <emix:marketContext>http://MijeongPark-PC:8080/oadr2-vtn-
groovy/program/a</emix:marketContext>
```

```
<oadrDistributeEvent xmlns="http://openadr.org/oadr-2.0a/2012/07"
xmlns:emix="http://docs.oasis-open.org/ns/emix/2011/06"
xmlns:pyld="http://docs.oasis-open.org/ns/energyinterop/201110/payloads"
xmlns:ei="http://docs.oasis-open.org/ns/energyinterop/201110"
xmlns:strm="urn:ietf:params:xml:ns:icalendar-2.0:stream"
xmlns:ical="urn:ietf:params:xml:ns:icalendar-2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><ei:eiResponse>
<ei:responseCode>404</ei:responseCode><pyld:requestID>2de2f123-2a85-4634-8a8d-
e0c5960798ad</pyld:requestID></ei:eiResponse><pyld:requestID>0701de91-24d4-4c76-
b530-731faa7d0d1d</pyld:requestID><ei:vtnID>ENOCtestVTN1</ei:vtnID>
</oadrDistributeEvent>
```

Fig. 4. The example of the wrong input value

We can prevent the wrong intentionally made input value as shown in Figure 4 through the specific pattern that can be used as input values as Figure 5. Therefore it needs to establish the specific pattern for Smart Grid and to perform the validation of input values.



```
If(!Pattern.matches("[0-9]+", val)) ...
```

Fig. 5. The example of the specific pattern

We verified the importance of complying with 'IDS00-J(Sanitize untrusted data passed across a trust boundary)' rule when utilizing open source software of Open-ADR based on the web environment by setting up the test environment. In this respect, verification of input values is needed, which can be implemented by restriction using the specific pattern.

4 Conclusion

This paper analyzed security weaknesses in open source software of OpenADR, which is DR protocol for demand reduction of Smart Grid. We derived several secure weaknesses of OpenADR, which violate secure coding rules according to CERT Java, by using the LDRA Testbed tool. In addition, we conducted experiments on the web environment in order to detect secure weaknesses, which could be found when we utilize the LDRA Testbed tool. As a result, we drive several secure weaknesses that allow malicious data modification and wrong input values. We can prevent economic and national loss through the software satisfying the functionality and the security by eliminating the security weaknesses.

As future work, we plan to establish new secure coding rules of Smart Grid for eliminating weaknesses. Also, we will perform the modeling of OpenADR protocol at its designing stage.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012R1A1A2009354). This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (NIPA-2014-H0301-14-1023) supervised by the NIPA(National IT Industry Promotion Agency).

References

1. David Kushner, The Real Story of Stuxnet. IEEE Spectrum, 2013.
2. CVE, CVE-2014-0160. CVE.MITRE, 2014.

3. Joe McManus MGR, The CERT Oracle Secure Coding Standard for Java. CERT, 2014.
4. Michael Hennell, LDRA Testbed and TBvision. LDRA, 1975.
5. OpenADR Alliance, OpenADR 2.0 Profile Specification A Profile. OpenADR Alliance, 2011.
6. CWE, Common Weakness Enumeration. CWE.MITRE, 1999.
7. LDRA, LDRA Getting Started Tutorial. LDRA Software Technology.
8. Charles McParland, OpenADR Open Source Toolkit: Developing Open Source Software for the Smart Grid. IEEE Power & Energy Society General Meeting, 2011.
9. CERT, OBJ01-J. Declare data members as private and provide accessible wrapper methods. CERT, 2012.
10. CWE, CWE-766: Critical Variable Declared Public. CWE.MITRE, 2014.
11. CVE, CVE-2010-3860. CVE.MITRE, 2010.
12. CERT, EXP00-J. Do not ignore values returned by methods. CERT, 2014.
13. CWE, CWE-252: Unchecked Return Value. CWE.MITRE, 2014.
14. CVE, CVE-2010-0211. CVE.MITRE, 2010.