Research & Technique

웹로직 원격 코드 실행 취약점 분석(CVE-2019-2729)

취약점 개요

지난 4월 26일 Oracle에서 제공하는 Java EE 기반의 웹 애플리케이션 서버(Web Application Server, WAS)인 웹로직에서 원격 코드 실행 취약점(CVE-2019-2725)이 발견되었다. 공격자는 이 취약점을 악용하여 권한 없이 원격으로 임의의 명령을 실행하였다. 6월 19일에는 CVE-2019-2725 취약점에 대한 보안패치를 우회하여 공격할 수 있는 취약점(CVE-2019-2729)이 발견되었다. CVE-2019-2729 취약점은 웹로직이 JDK 1.6 버전에서 구동될 시 필터링 우회가 가능하여 발생하였다. 이번 달 〈Research & Technique〉에서는 CVE-2019-2725 취약점과 해당 패치 소스, 그리고 CVE-2019-2729 취약점에 대해서 살펴보고자 한다.

영향 받는 소프트웨어 버전

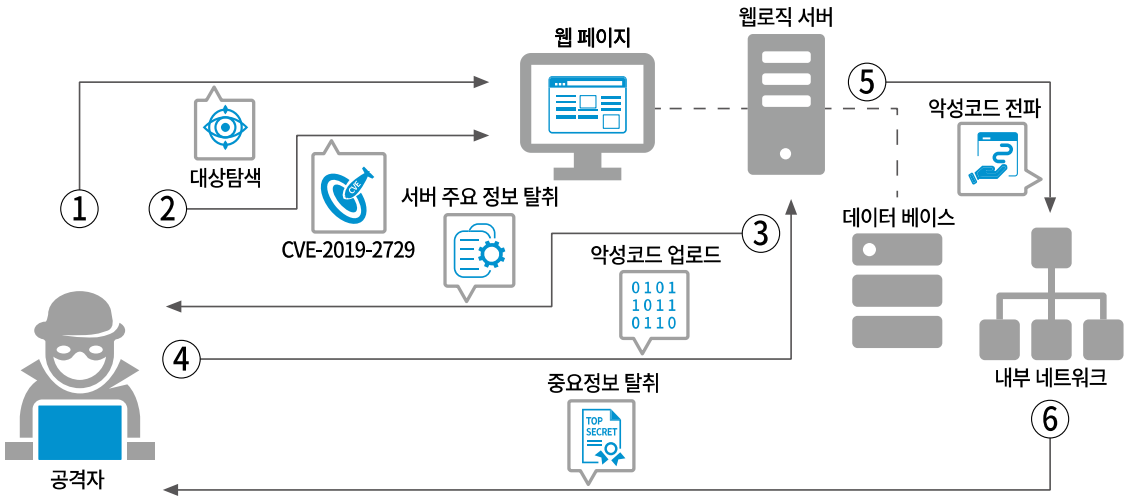
S/W 구분	취약 버전
WebLogic Server	10.3.6.0 12.1.3.0 12.1.3.0 } +CVE-2019-2725 patch

테스트 환경 구성 정보

역할 구분	정보
공격자	Windows 10 64bit
희생자	Windows 10 64bit / WebLogic Server 10.3.6 / JDK 1.6

공격 시나리오

웹로직 CVE-2019-2729 취약점을 이용하여 취약한 대상의 정보를 탈취하는 시나리오는 다음과 같다.



[공격 시나리오]

① 공격자는 운영 중인 페이지가 취약한 버전의 웹로직 서버를 사용하는 것을 확인함

② 공격자는 취약한 페이지 요청을 변조하여 서버에서 악의적인 명령이 실행되도록 함

③ 공격자는 이를 통해 웹 서버 내의 주요 정보(시스템 설정파일, 소스코드)들을 획득함

④ 공격자는 서버에 악성 코드를 업로드 함

⑤ 공격자는 악성코드를 통해 내부 네트워크에 침입함

⑥ 공격자는 지속적으로 내부 네트워크의 중요 정보를 탈취함

취약점 상세 분석

Step 1. CVE-2019-2725 취약점 분석

WebService 구성 요소에 대한 비동기 통신을 제공하는 wls9_async_response.war 모듈에서 CVE-2019-2725 취약점이 발견되었다. 웹로직은 Java Object를 직렬화를 통해 Byte로 변환한 후 XML로 Context를 전달하여 통신하는데, 공격자는 웹로직의 wls9_async_response.war 모듈에 포함된 /_async/AsyncResponseService 페이지에 접근하여 조작된 XML 데이터를 SOAP[1] 로 전달할 수 있다. 공격자는 XML데이터를 Byte 형식으로 〈work:WorkContext〉 태그로 감싸 전달한다. 이때 요청하는 SOAP 헤더 정보 중 RelatesTo 속성에 따라 해당 데이터의 처리를 결정한다. RelatesTo 속성이 설정되지 않으면 Byte로 변환된 데이터를 원래의 object나 data로 변환하는 역직렬화 과정을 진행하지 않는다.

AsyncResponseBean.class | AsyncResponseHandler.class

```
public boolean handleRequest(MessageContext paramMessageContext) {
  if (paramMessageContext == null) {
    return true;
  }
  if (!(paramMessageContext instanceof javax.xml.rpc.handler.soap.SOAPMessageContext))
  {
    return true;
  }

  if (verbose) {
    Verbose.log("AsyncResponseHandler.handleRequest");
  }
  String str1 = (String)paramMessageContext.getProperty("weblogic.wsee.addressing.RelatesTo");


  if (str1 == null) {
    return false;
  }

  String str2 = "Unknown";
  String str3 = "";
  if (verbose) {
    MsgHeaders msgHeaders = ((SoapMessageContext)paramMessageContext).getHeaders();
    MessageIdHeader messageIdHeader = (MessageIdHeader)msgHeaders.getHeader(MessageIdHeader.TYPE);
    str2 = (messageIdHeader != null) ? messageIdHeader.getMessageId() : "Unknown";
    SequenceHeader sequenceHeader = (SequenceHeader)msgHeaders.getHeader(SequenceHeader.TYPE);
    if (sequenceHeader != null) {
      str3 = " on reliable sequence " + sequenceHeader.getSequenceId() + " and seq num " + sequenceHeader.getMessageNumber();
    } else {
      String str4 = (String)paramMessageContext.getProperty("weblogic.wsee.reliability.RequestMessageSeqNumber");
      String str5 = (String)paramMessageContext.getProperty("weblogic.wsee.reliability.RequestMessageSeqID");
      if (str5 != null) {
        str3 = " on reliable sequence " + str5 + " and seq num " + str4;
      }
    }
    Verbose.log("Processing async response message " + str2 + " related to request msg " + str1 + str3);
    StringBuffer stringBuffer = HandlerIterator.getHandlerHistory(paramMessageContext);
    if (stringBuffer != null) {
      Verbose.log("Async response message " + str2 + " related to request msg " + str1 + str3 + " has this handler history: " + stringBuffer);
    }
  }
```

[RelatesTo 속성 확인 코드]

[1] Simple Object Access Protocol: HTTP, HTTPS, SMTP 등을 통해 XML 기반의 메시지를 컴퓨터 네트워크 상에서 교환하는 프로토콜이다.

다음 PoC 코드는 SOAP 헤더 정보를 작성하고, RelatesTo 속성을 임의로 설정하여 전달한 내용이 역직렬화를 진행하도록 한다.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:asy="http://www.bea.com/async/AsyncResponseService">
<soapenv:Header>
<wsa:Action>xxx</wsa:Action>
<wsa:RelatesTo>xxx</wsa:RelatesTo>
<work:WorkContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
```

이후 wls9_async_response.war 모듈은 전달받은 Context 정보를 XMLDecoder를 이용해 역직렬화한다. 이때, WorkContextXmlInputAdapter.class에 정의된 validate() 함수에서 원격 코드 실행에 사용 가능한 〈object〉, 〈new〉, 〈method〉 태그들을 블랙리스트로 검증한다.

WorkContextXmlInputAdapter.class

```
60  private void validate(InputStream is) {
      WebLogicSAXParserFactory factory = new WebLogicSAXParserFactory();

64    try {
        SAXParser parser = factory.newSAXParser();
65      parser.parse(is, new DefaultHandler() {
66            private int overallarraylength = 0;

          public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
72          if (qName.equalsIgnoreCase("object")) {
73            throw new IllegalStateException("Invalid element qName:object");
            }

77          if (qName.equalsIgnoreCase("new")) {
78            throw new IllegalStateException("Invalid element qName:new");
            }

82          if (qName.equalsIgnoreCase("method")) {
83            throw new IllegalStateException("Invalid element qName:method");
            }
```

[취약점 발생 위치 확인1]

또한, 〈void〉 태그 뒤에는 〈index〉 태그만 사용할 수 있고, 〈array〉 태그의 class 속성은 byte 형식으로만 사용할 수 있도록 검증한다.

WorkContextXmlInputAdapter.class

```
82          if (qName.equalsIgnoreCase("method")) {
83            throw new IllegalStateException("Invalid element qName:method");
            }


88          if (qName.equalsIgnoreCase("void")) {
89            for (int i = 0; i < attributes.getLength(); i++) {
90              if (!"index".equalsIgnoreCase(attributes.getQName(i))) {
91                throw new IllegalStateException("Invalid attribute for element void:" + attributes.getQName(i));
                }
              }
            }


97          if (qName.equalsIgnoreCase("array"))
            {
99            String attClass = attributes.getValue("class");
100           if (attClass != null && !attClass.equalsIgnoreCase("byte")) {
101             throw new IllegalStateException("The value of class attribute is not valid for array element.");
              }


105           String lengthString = attributes.getValue("length");
106           if (lengthString != null) {
                try {
108               int length = Integer.valueOf(lengthString).intValue();

110               if (length >= MAXARRAYLENGTH) {
111                 throw new IllegalStateException("Exceed array length limitation");
                  }
```

[취약점 발생 위치 확인2]

CVE-2019-2725 취약점은 위의 블랙리스트를 우회하기 위해 〈class〉 태그를 사용한다. 〈class〉 태그는 클래스의 인스턴스를 나타내며, 〈class〉 태그를 사용하면 모든 클래스의 인스턴스를 생성할 수 있다. 다음 PoC 코드는 〈class〉 태그를 이용하여 UnitOfWorkChangeSet 클래스를 실행한다. 여기서 사용하는 UnitOfWorkChangeSet 클래스는 인수로 〈array class〉 태그 내에 존재하는 byte 배열을 받아 역직렬화하기 때문에 악의적으로 삽입된 코드를 실행할 수 있다.

```
〈java〉
〈class〉
〈string〉oracle.toplink.internal.sessions.UnitOfWorkChangeSet〈/string〉
〈void〉
〈array class="byte" length="3478"〉
〈void index="0"〉
〈byte〉-84〈/byte〉
〈/void〉
```

Step 2. CVE-2019-2729 취약점 분석

CVE-2019-2729 취약점은 CVE-2019-2725 취약점에서 사용한 〈class〉 태그를 사용할 수 없어 공격자는 〈class〉 태그를 이용해 원격으로 임의의 코드를 실행할 수 없다. 하지만 JDK 1.6버전에서 XMLDecoder 시 〈array method="forName"〉 태그로 〈class〉 태그를 대체하여 사용할 수 있다. 따라서 〈class〉 태그를 〈array method = "forName"〉 태그로 변경하면, 〈class〉 태그를 사용하지 않고 임의의 클래스를 실행할 수 있다.

```
〈java〉
〈array method="forName"〉
〈string〉oracle.toplink.internal.sessions.UnitOfWorkChangeSet〈/string〉
〈void〉
〈array class="byte" length="5010"〉
〈void index="0"〉
〈byte〉-84〈/byte〉
〈/void〉
```

Step 3. CVE-2019-2725 취약점 패치 소스코드 분석

Oracle에서는 CVE-2019-2725 취약점에 사용된 〈class〉 태그를 블랙리스트에 추가해 공격자가 클래스의 인스턴스를 생성하지 못하게 하였고, 이를 통해 삽입된 코드가 실행되지 않도록 수정하였다.

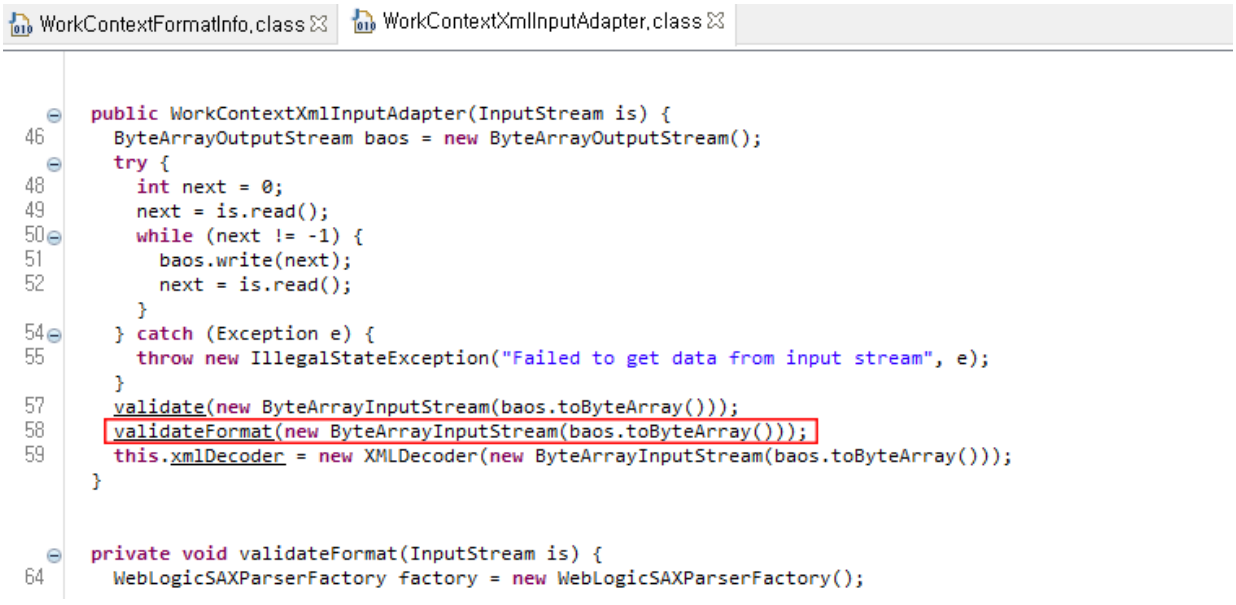
WorkContextXmlInputAdapter.class

```
60  private void validate(InputStream paramInputStream) {
      WebLogicSAXParserFactory webLogicSAXParserFactory = new WebLogicSAXParserFactory();

64    try {
        SAXParser sAXParser = webLogicSAXParserFactory.newSAXParser();
65      sAXParser.parse(paramInputStream, new DefaultHandler() {
66            private int overallarraylength = 0;



            public void startElement(String param1String1, String param1String2, String param1String3, Attributes param1Attributes) throws SAXException
72            if (param1String3.equalsIgnoreCase("object")) {
73              throw new IllegalStateException("Invalid element qName:object");
              }

76            if (param1String3.equalsIgnoreCase("class")) {
77              throw new IllegalStateException("Invalid element qName:class");
              }

81            if (param1String3.equalsIgnoreCase("new")) {
82              throw new IllegalStateException("Invalid element qName:new");
              }

86            if (param1String3.equalsIgnoreCase("method")) {
87              throw new IllegalStateException("Invalid element qName:method");
              }


92            if (param1String3.equalsIgnoreCase("void")) {
93              for (byte b = 0; b < param1Attributes.getLength(); b++) {
94                if (!"index".equalsIgnoreCase(param1Attributes.getQName(b))) {
95                  throw new IllegalStateException("Invalid attribute for element void:" + param1Attributes.getQName(b));
                  }
                }
              }
```

[CVE-2019-2725 보안패치 확인]

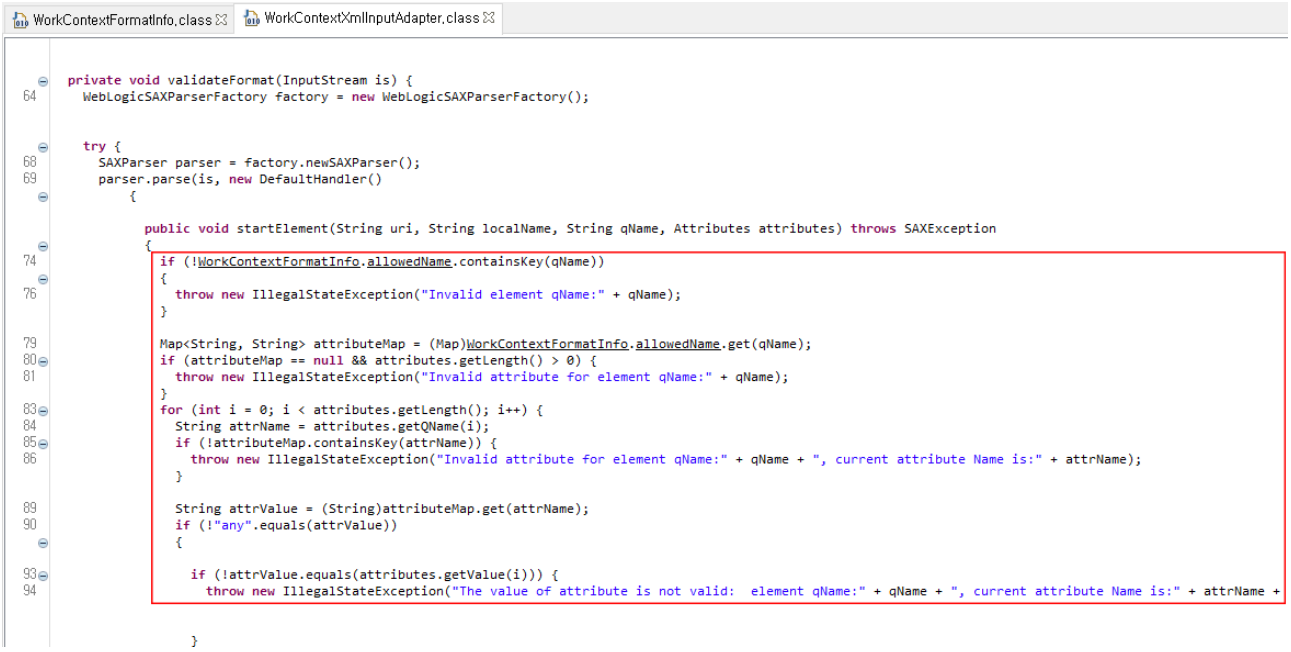
Step 4. CVE-2019-2729 취약점 패치 소스코드 분석

추가된 CVE-2019-2729 보안패치는 이전 패치와 달리 블랙리스트가 아닌 화이트리스트 방식으로 데이터를 검증하여 필요한 태그 외에는 사용할 수 없도록 수정되었다.
먼저 WorkContextXmlInputAdapter.class에서 기존 validate() 함수 외에 validateFormat() 함수를 추가하여 XML 데이터를 확인한다.



[XML 데이터 추가 검증 함수 확인]

추가된 validateFormat() 함수에서 전달받은 XML 데이터가 WorkContextFormatInfo.class에 정의되었는지 확인한다.



[화이트리스트 적용 코드]

WorkContextFormatInfo.class에 사용할 수 있도록 허용된 내용은 다음과 같다.

WorkContextFormatInfo.class | WorkContextXmlInputAdapter.class

```
package weblogic.wsee.workarea;

import java.util.HashMap;
import java.util.Map;


public class WorkContextFormatInfo
{
  public static final Map<String, Map<String, String>> allowedName = new HashMap();

  static  {
    allowedName.put("string", null);
    allowedName.put("int", null);

    allowedName.put("long", null);

    allowedAttr = new HashMap();
    allowedAttr.put("class", "byte");
    allowedAttr.put("length", "any");

    allowedName.put("array", allowedAttr);

    allowedAttr = new HashMap();
    allowedAttr.put("index", "any");

    allowedName.put("void", allowedAttr);

    allowedName.put("byte", null);
    allowedName.put("boolean", null);
    allowedName.put("short", null);
    allowedName.put("char", null);
    allowedName.put("float", null);
    allowedName.put("double", null);

    allowedAttr = new HashMap();
    allowedAttr.put("class", "java.beans.XMLDecoder");
    allowedAttr.put("version", "any");

    allowedName.put("java", allowedAttr);
  }
}
```

[화이트리스트 확인]

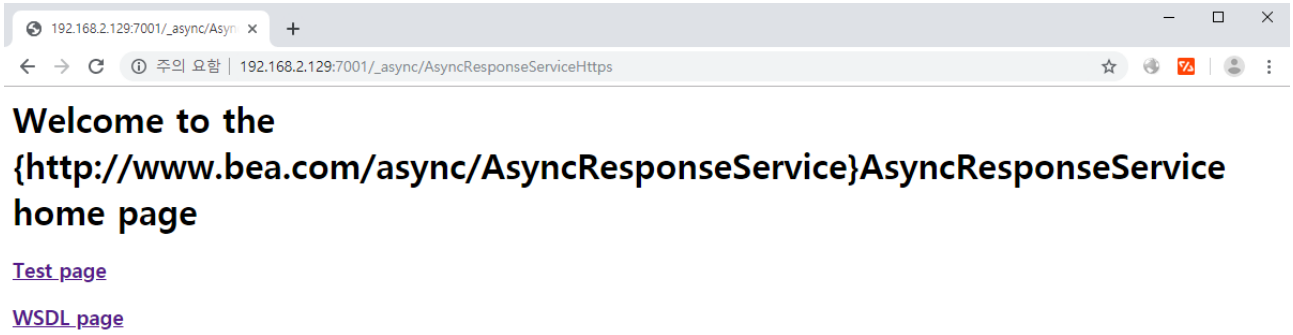
취약점 테스트

Step 1. 취약점 테스트 환경 구축

Windows 10 운영체제에 JDK 1.6 버전을 설치하고 취약한 버전의 웹로직(10.3.6)을 다운로드하여 테스트 PC에 설치한다. 그리고 CVE-2019-2729 취약점을 테스트하기 위해 CVE-2019-2725 보안패치를 적용한다. 설치한 웹로직 실행파일인 'startWebLogic.cmd'를 실행시켜 웹로직이 구동되도록 한다.

웹로직 실행 파일 기본 경로 : ~₩Oracle₩Middleware₩user_projects₩domains₩base_domain. . .

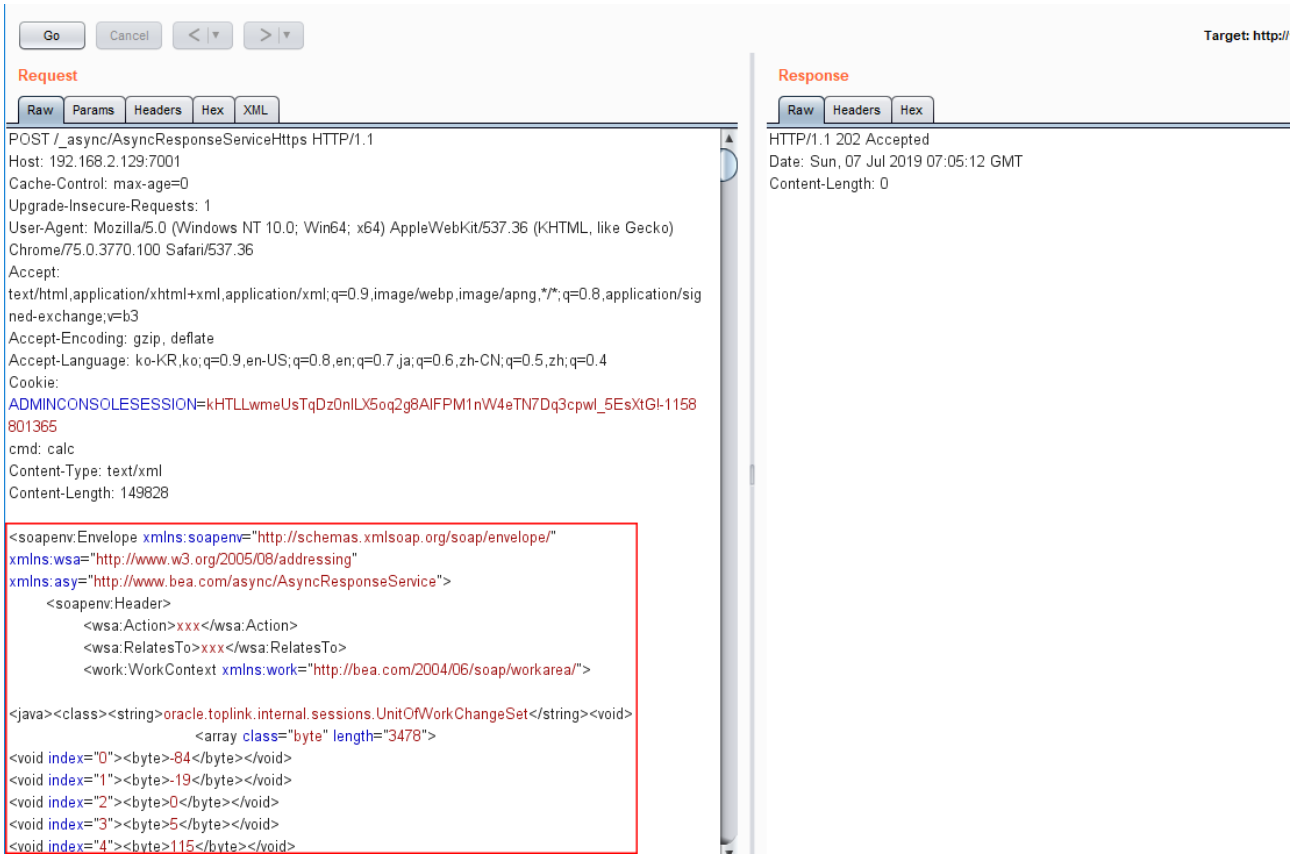
관리자 콘솔을 확인하여 웹로직이 구동되고 있으면 CVE-2019-2729 취약점이 존재하는 주소 'HTTP://희생자IP:7001/_async/AsyncResponseServiceHttps'에 접근한다.



[취약한 페이지 접근]

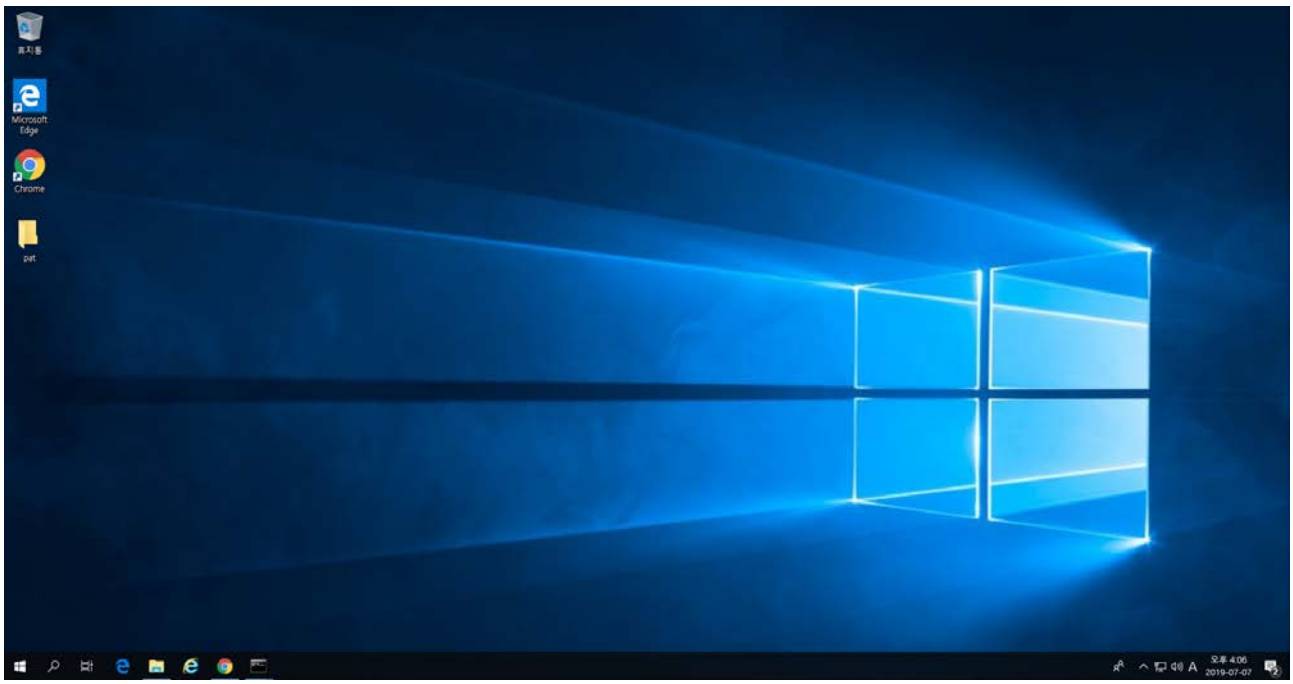
Step 2. PoC 테스트

취약한 페이지에 접근할 때 희생자 PC에서 계산기를 실행하는 CVE-2019-2725 취약점 PoC 코드를 삽입한다.



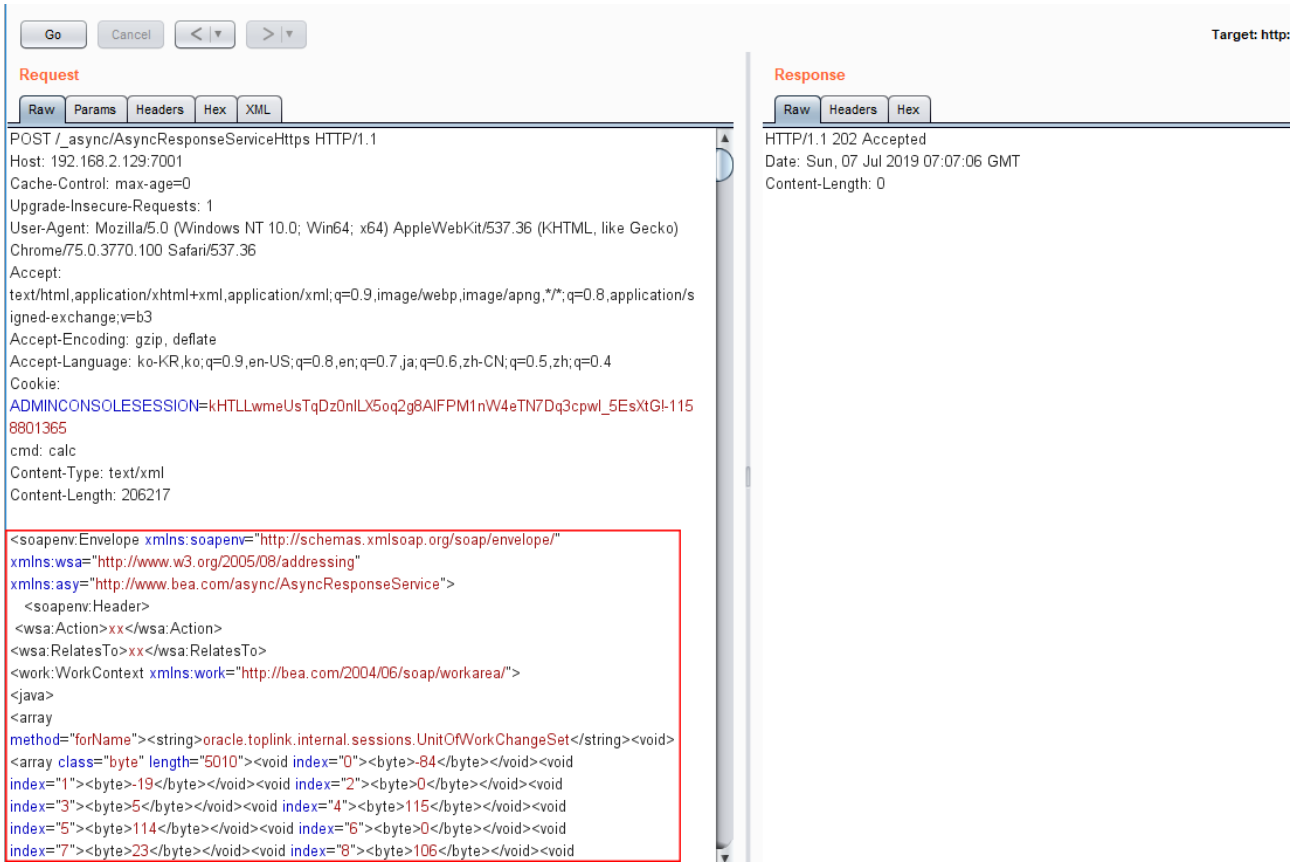
[CVE-2019-2725 PoC코드 삽입]

그러면 계산기가 실행되지 않고 CVE-2019-2725 취약점 패치가 정상적으로 적용된 것을 알 수 있다.



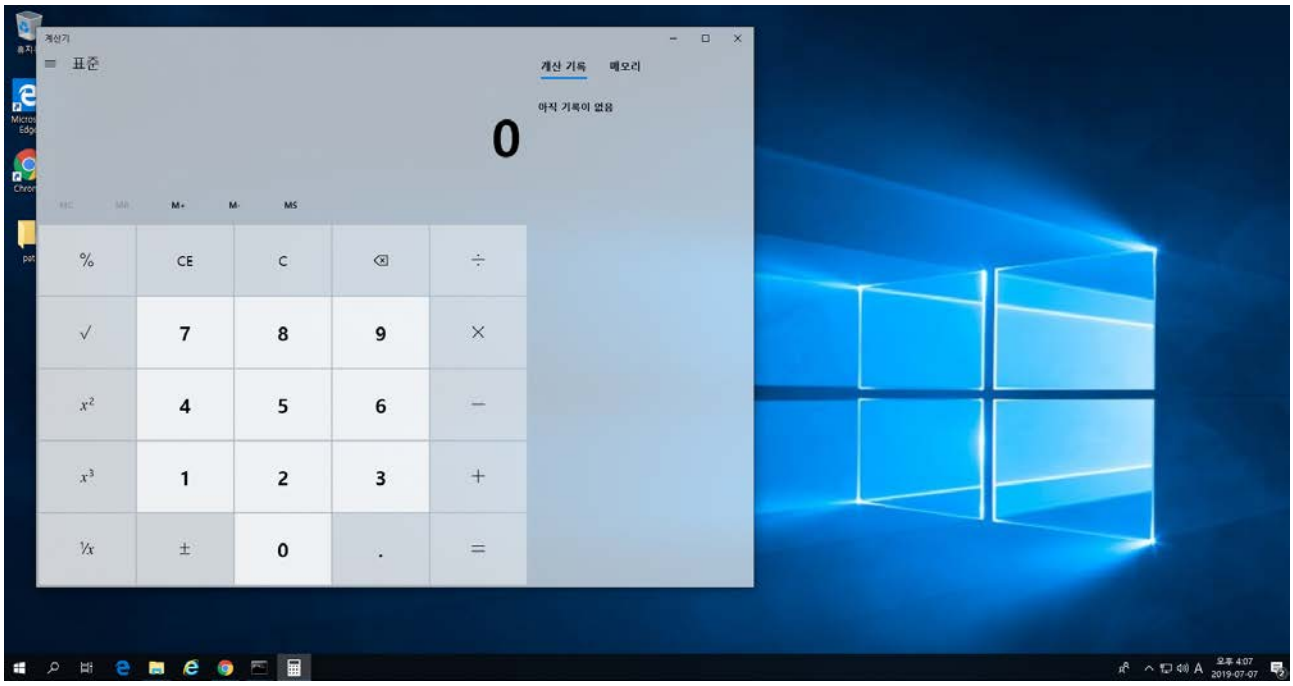
[공격실패 확인]

이후 CVE-2019-2729 취약점 PoC 코드를 삽입한다.



[CVE-2019-2729 PoC코드 삽입]

그러면 계산기가 실행되고 CVE-2019-2729 취약점 공격이 성공한 것을 알 수 있다.



[공격성공 확인]

대응 방안

Oracle에서 발표한 웹로직 보안패치를 다운로드하여 적용한다.
보안패치 적용이 어려운 경우 취약점이 발생하는 wls9_async_response.war, wls-wsat.war를 삭제하거나, /_aysnc/* 및 /wls-wsat/* 경로에 대해 적절한 접근 통제가 필요하다.
https://www.oracle.com/technetwork/security-advisory/alert-cve-2019-2729-5570780.html

Oracle Security Alert Advisory - CVE-2019-2729

Description

This Security Alert addresses CVE-2019-2729, a deserialization vulnerability via XMLDecoder in Oracle WebLogic Server Web Services. This remote code execution vulnerability is remotely exploitable without authentication, i.e., may be exploited over a network without the need for a username and password.

Due to the severity of this vulnerability, Oracle strongly recommends that customers apply the updates provided by this Security Alert as soon as possible.

Affected Products and Patch Information

Security vulnerabilities addressed by this Security Alert affect the products listed below. The product area is shown in the Patch Availability Document column. Please click on the links in the Patch Availability Document column below to access the documentation for patch availability information and installation instructions.

Affected Products and Versions	Patch Availability Document
Oracle WebLogic Server, versions 10.3.6.0.0, 12.1.3.0.0, 12.2.1.3.0	Fusion Middleware

[웹로직 취약점 패치]