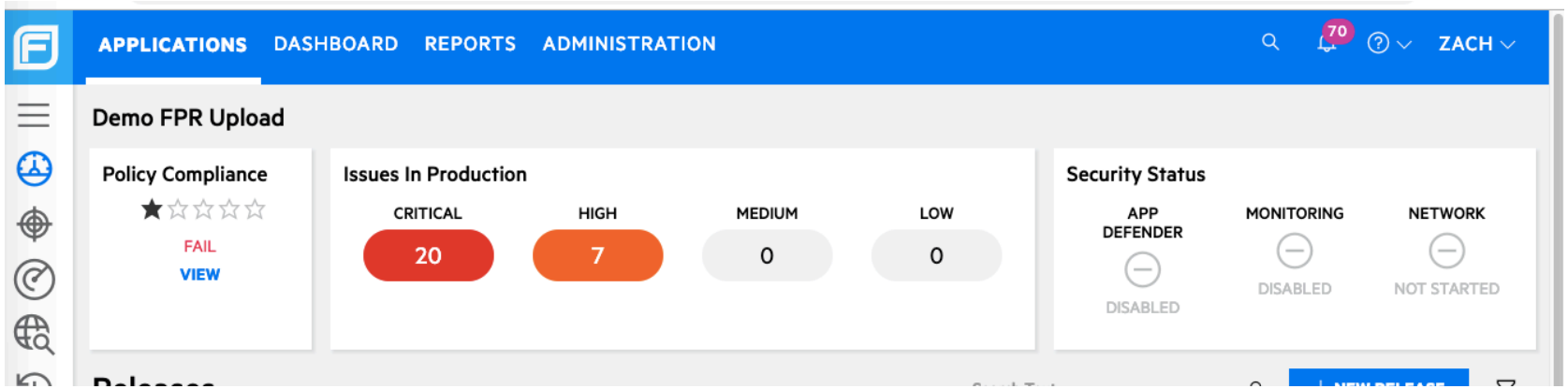
FOD Quick start



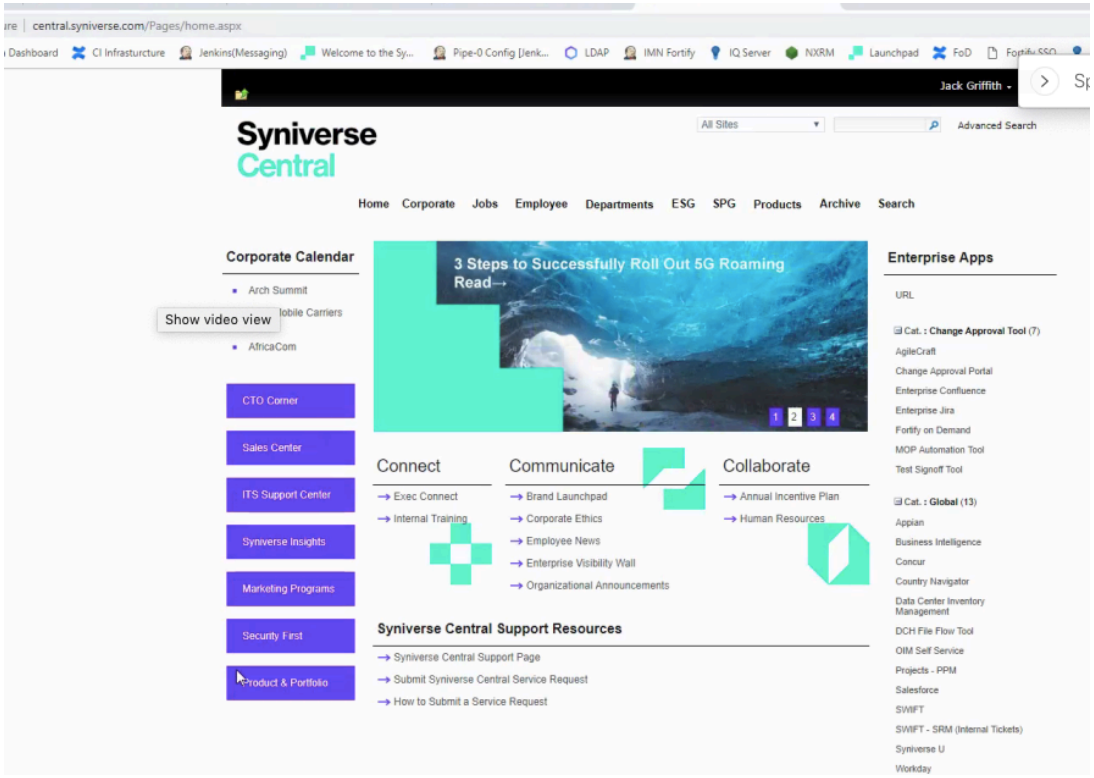
Agenda

1. How to log in
2. How to upload code
3. How to initiate a scan
4. How to check progress of a scan
5. How to view results (as a whole set)
6. How to sort, prioritize, parse results
7. How to dive deeper into individual results
8. How to create reports
9. How to use the Security Assistant
10. How to access Secure Code Warrior

How to log in

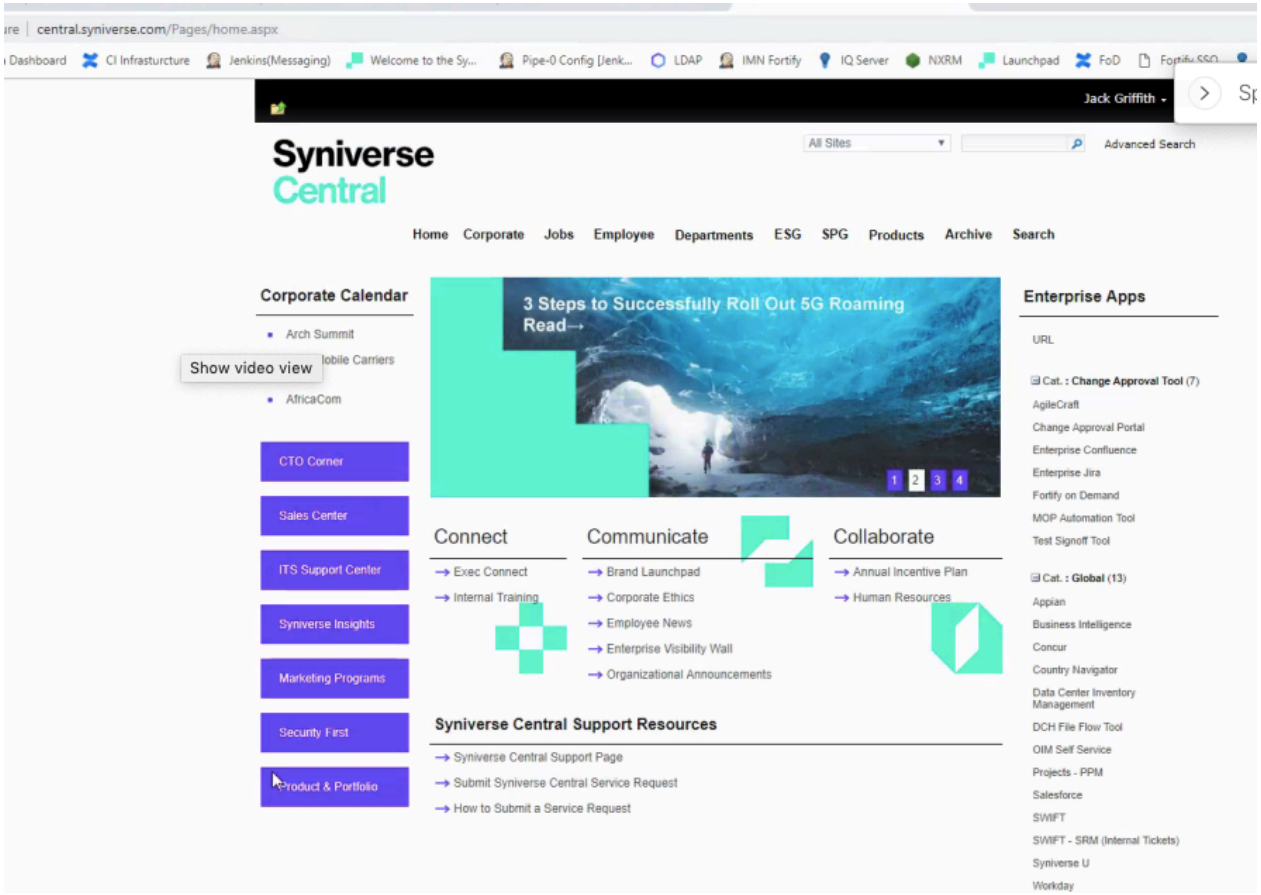
NOTE: You should have received an email with you login information for user, password and tenant.

Open Web Browser and go to https://central.syniverse.com



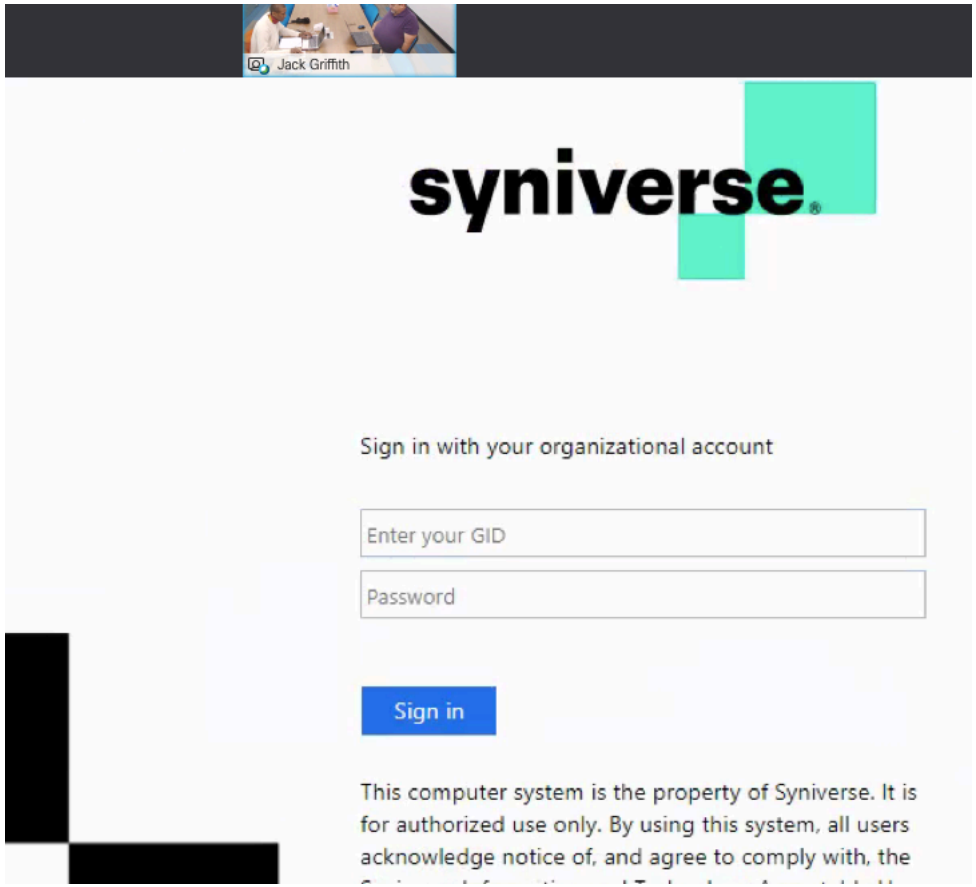
How to log in - 2

Click Fortify on Demand Link



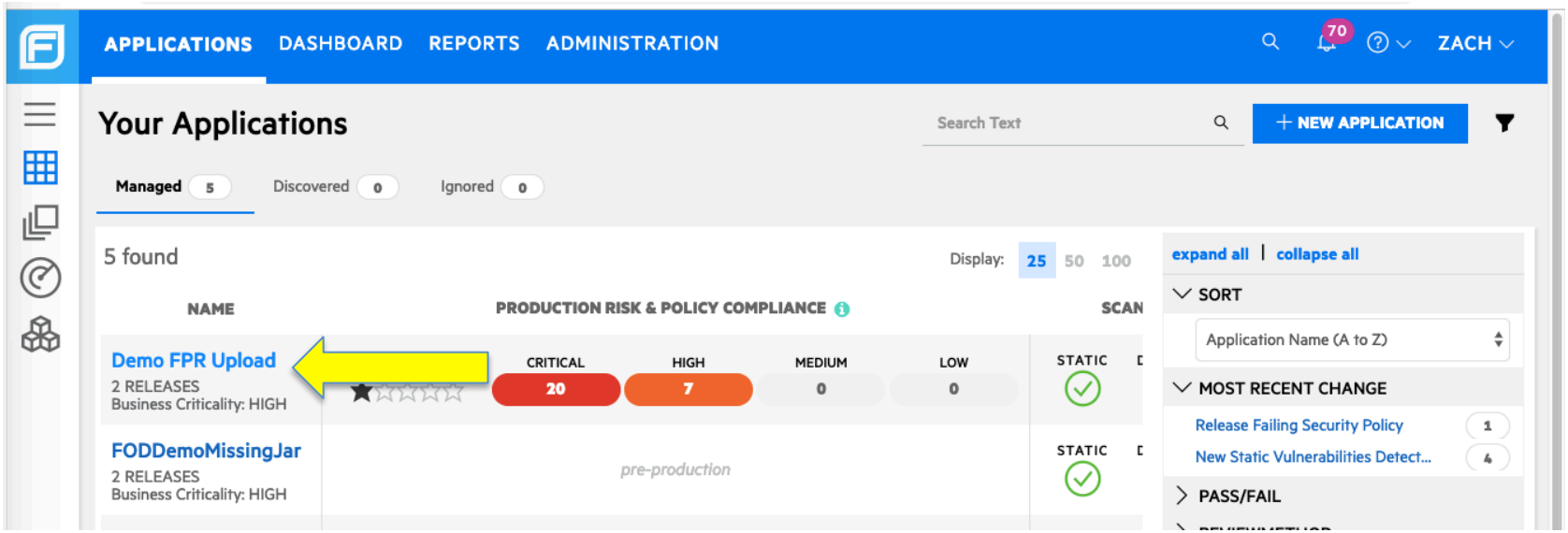
How to log in - 3

Login with your SSO



Understanding the Dashboard

Main Dashboard Screen



Preparing Java Zip for Upload

For Java applications, package the compiled and source code files for upload to Fortify on Demand as follows.

Preparing Compiled Files (Required)

Compile all files in debug mode. If they are not, the assessment will still be completed, but the results will not include line numbers of issues.

Ensure that all third-party JAR files are included. This prevents missing dependency issues from appearing during assessment.

Web application: Package in a WAR or EAR file. You can package multiple WAR files as an EAR file.

Non-web application: Ensure that there are no precompiled JSPs.

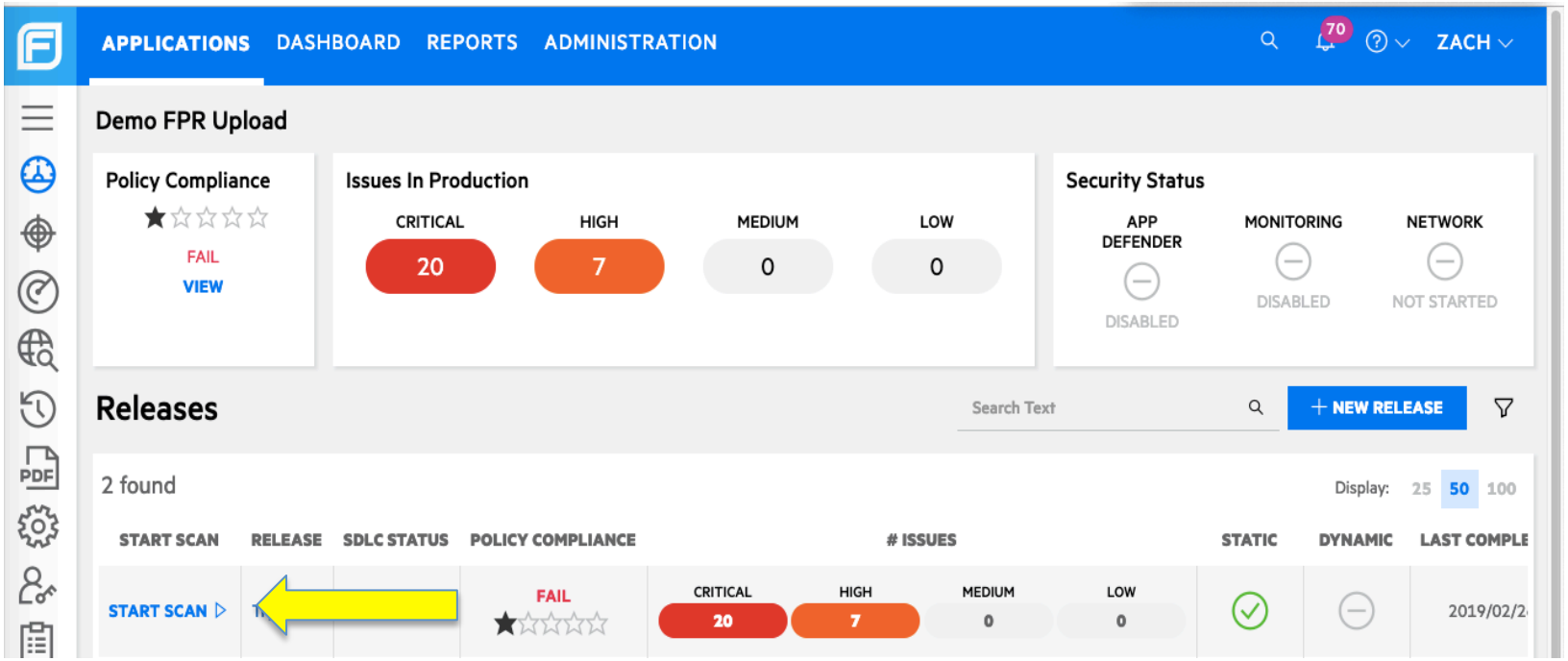
Package all application files in one zip file.

Preparing Java Zip for Upload -2

```
#!/bin/sh
#
# Simple Java ZIP Files and Jars for upload to FOD.
#

if [ "$#" -gt 1 ];
Then
  APP_SOURCE_DIR=$1
  APP_JAR_DIR=$2

  mkdir tmp_foduplоad

for appfiles in `find ${APP_SOURCE_DIR} -type f -exec file {} \; | grep "ASCII text" | cut -d: -f1`
do
  cp $appfiles tmp_foduplоad
done

for jar_files in `find ${APP_JAR_DIR} -name "*.jar" `
do
  cp ${jar_files} tmp_foduplоad
done

zip -r tmp_foduplоad.zip tmp_foduplоad

else

echo "usage fodZipJava <App Source Dir> <App Jar Dir>"

fi
```

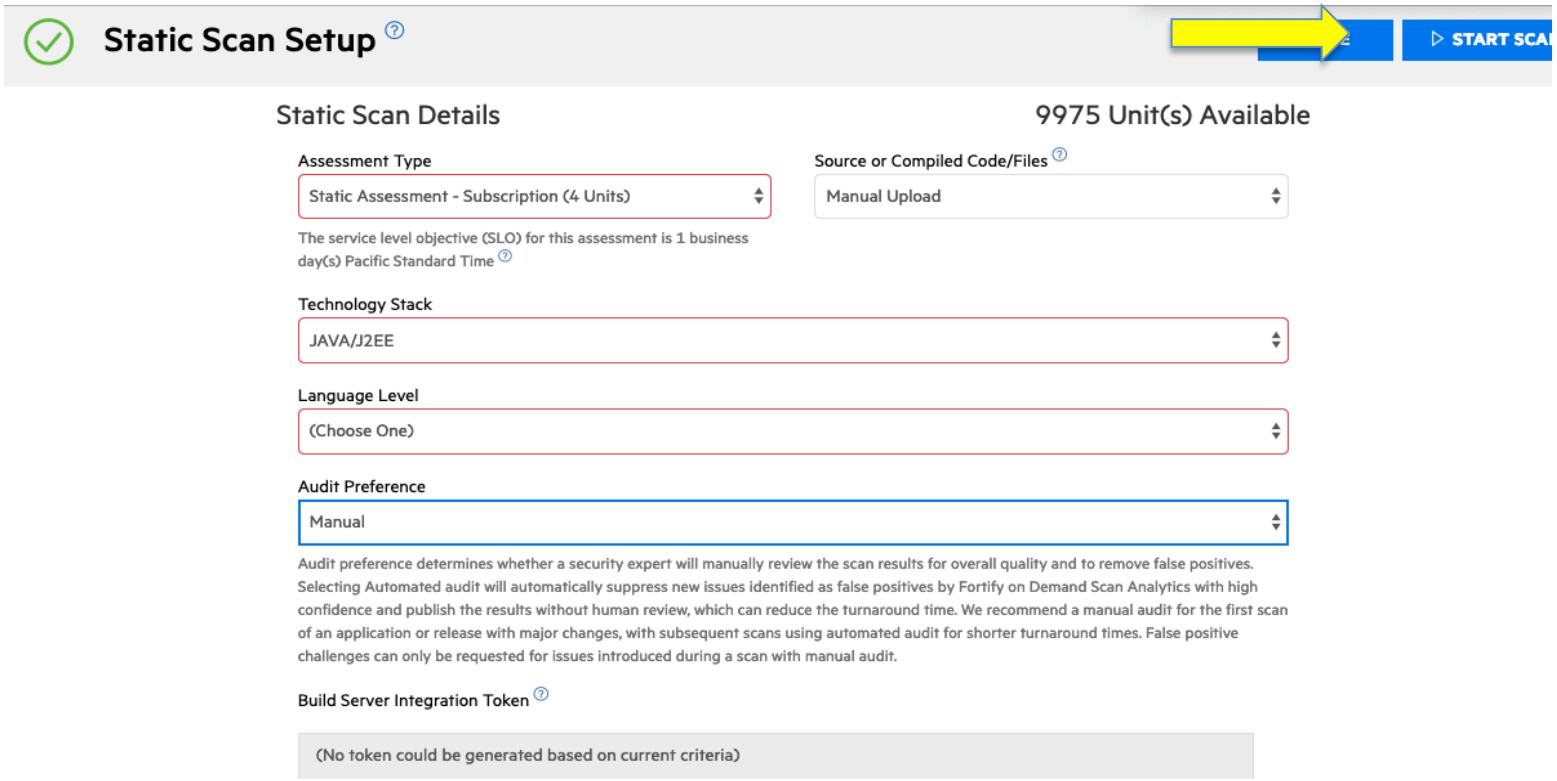
Start Scan - 1

Select Start Scan



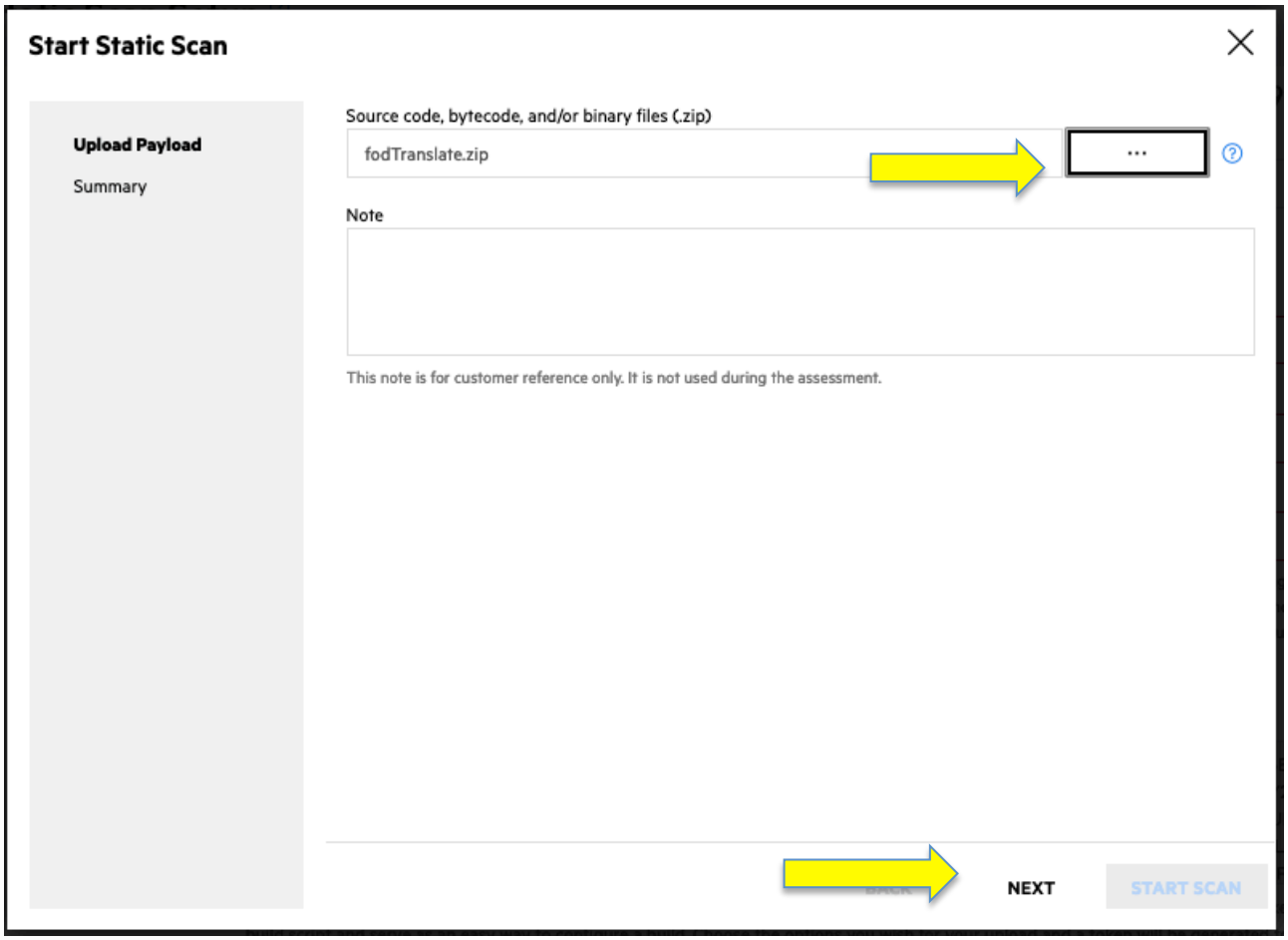
Start Scan - 2

Click Start Scan
NOTE: The DevOps team has already setup this information.



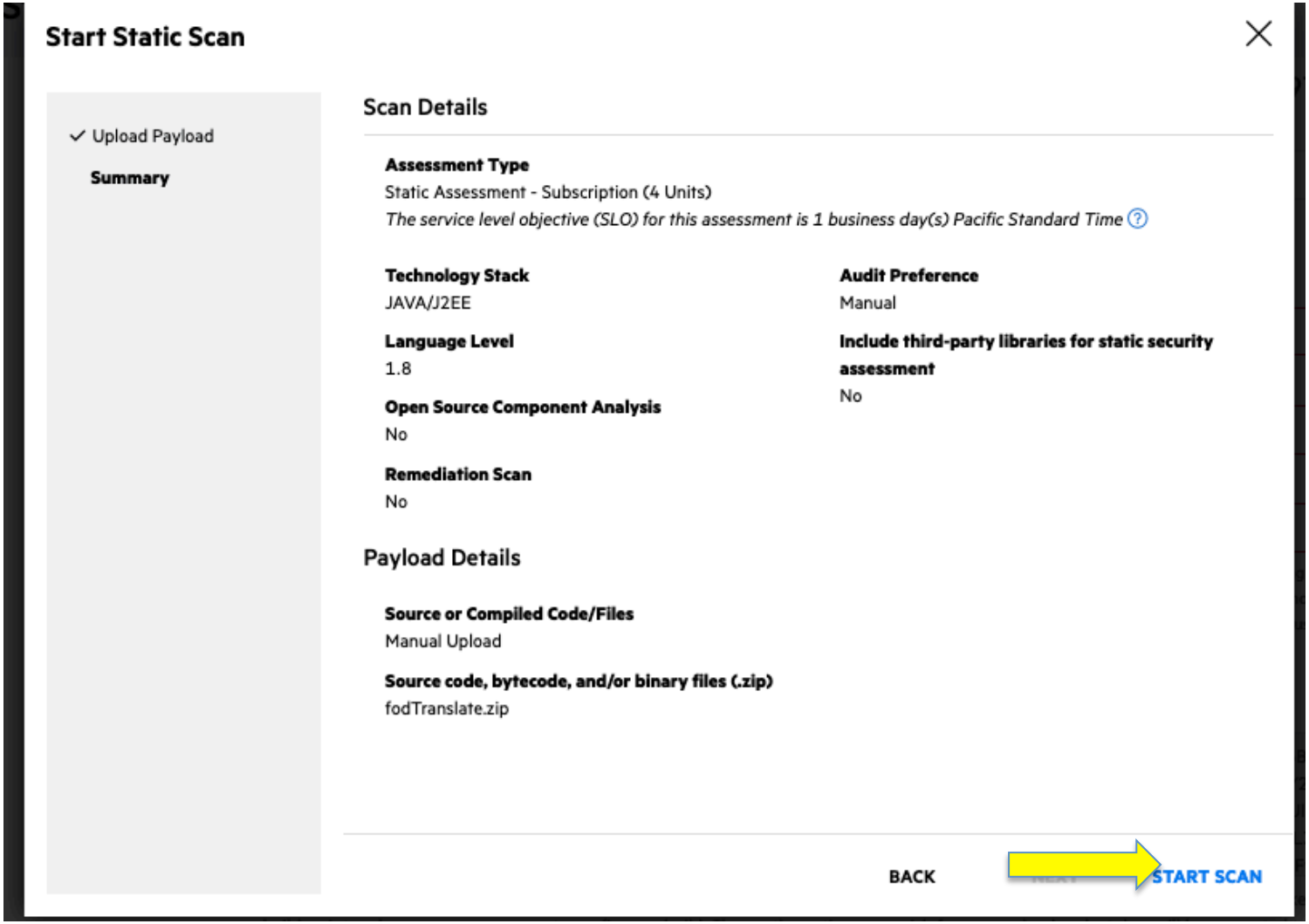
Start Scan - 3

Click … to Upload Zip File and then click next



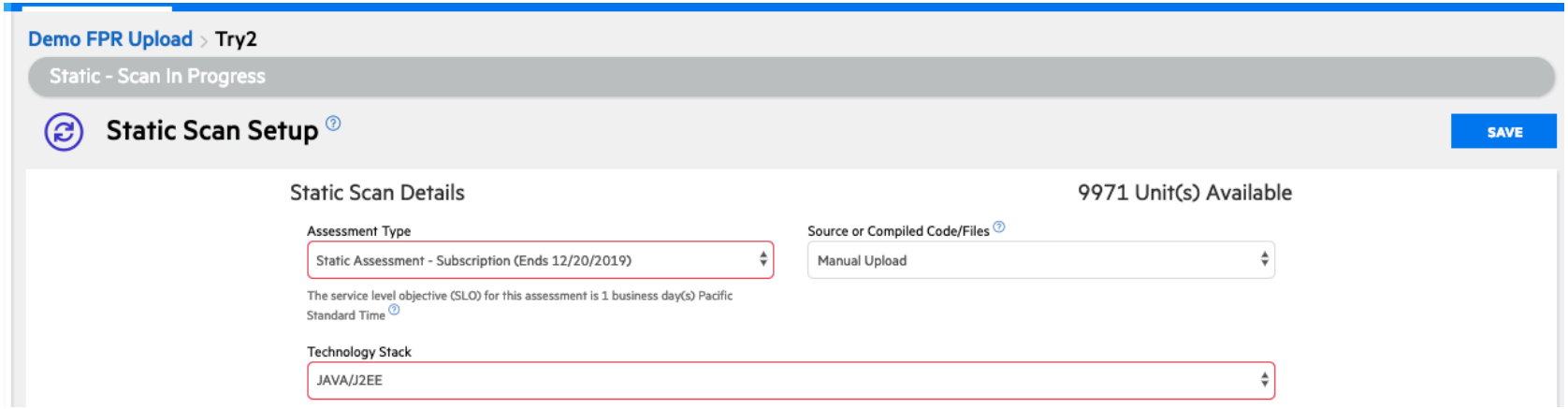
Start Scan - 4

Click Start Scan

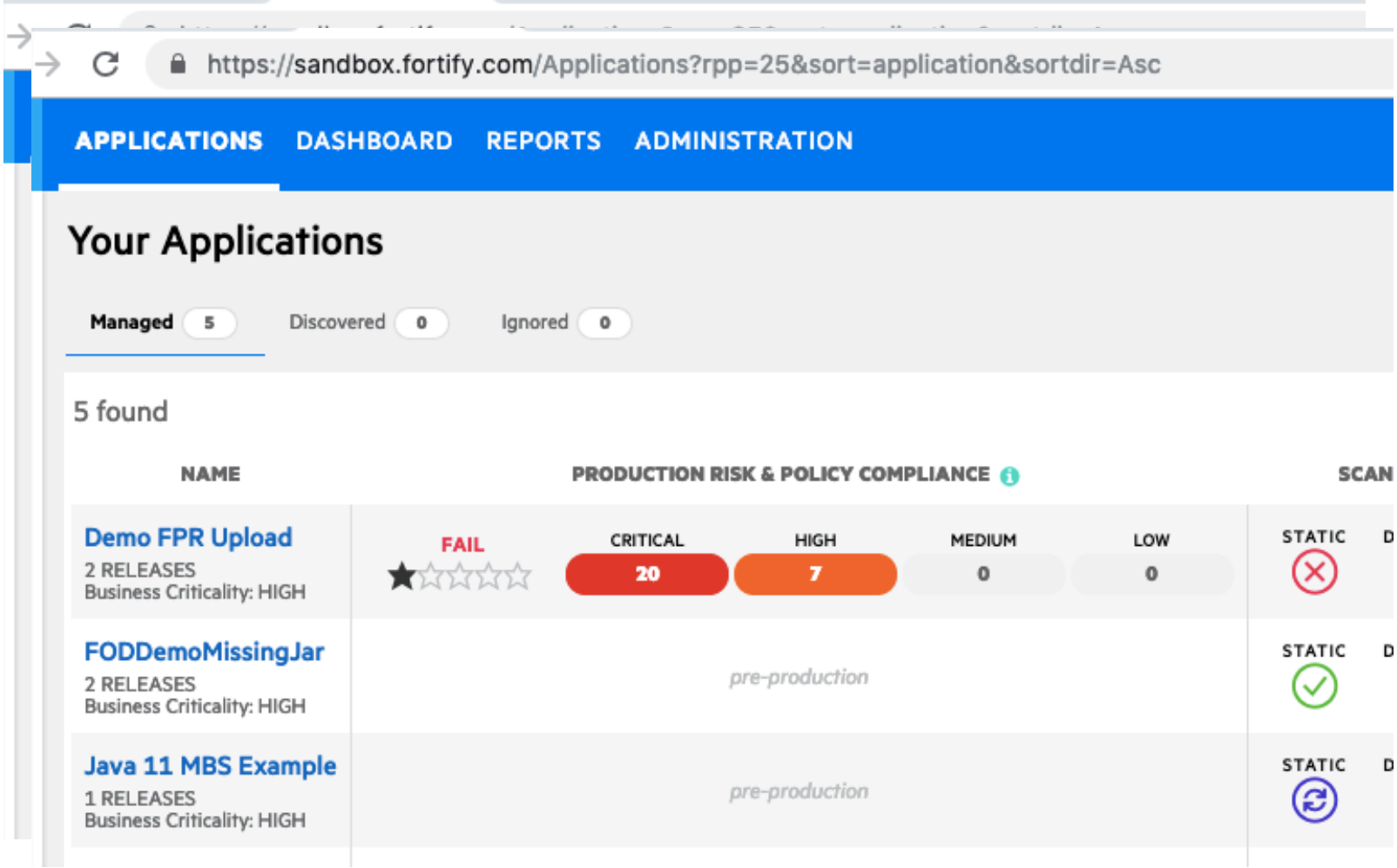


Start Scan - 5

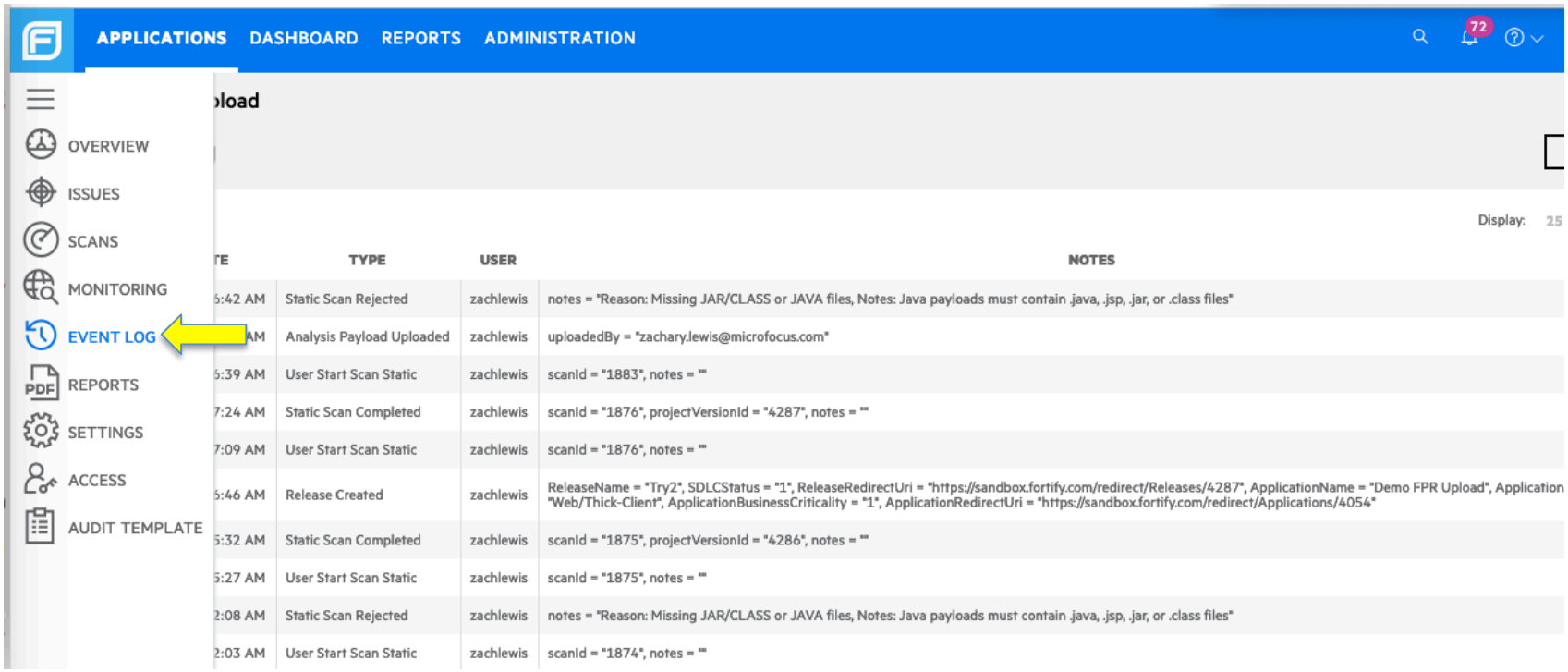
Scan in Progress



Check Status of Scan



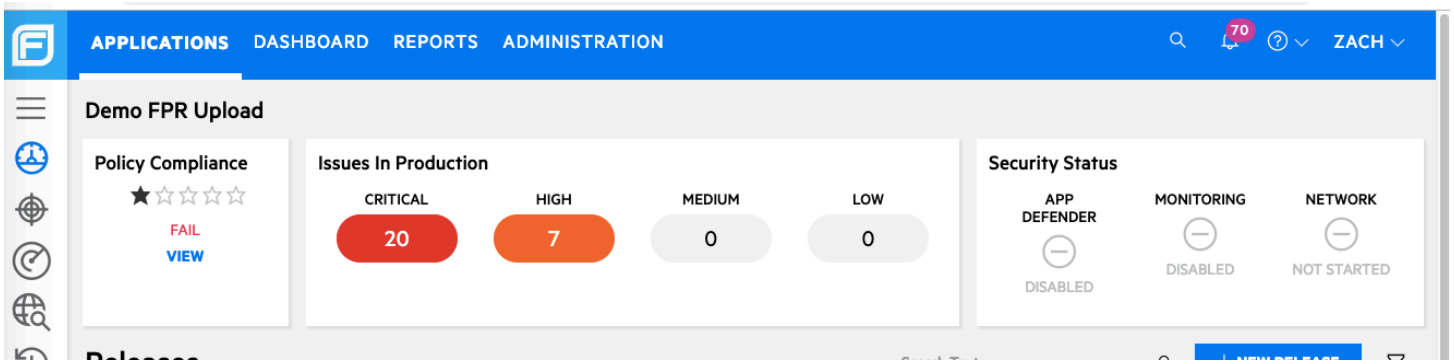
Check Status of Scan



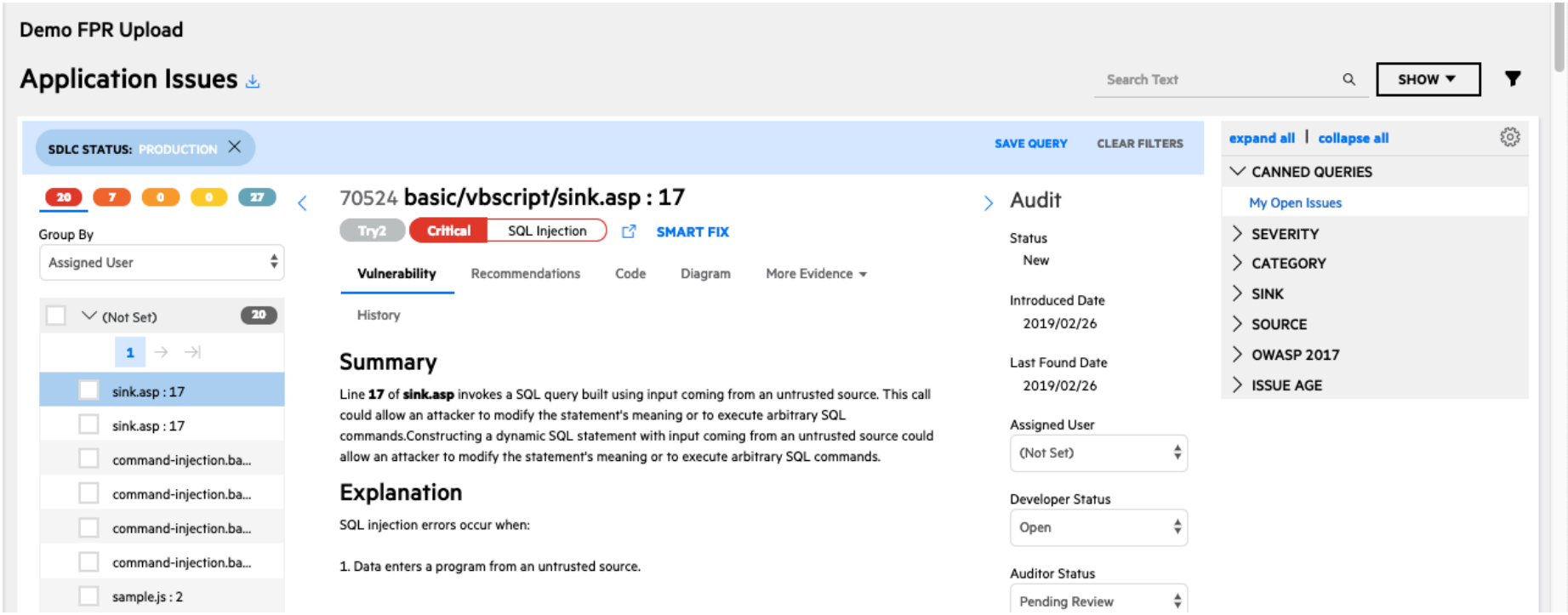
TE	TYPE	USER	NOTES
6:42 AM	Static Scan Rejected	zachlewis	notes = "Reason: Missing JAR/CLASS or JAVA files, Notes: Java payloads must contain .java, .jsp, .jar, or .class files"
AM	Analysis Payload Uploaded	zachlewis	uploadedBy = "zachary.lewis@microfocus.com"
6:39 AM	User Start Scan Static	zachlewis	scanId = "1883", notes = ""
7:24 AM	Static Scan Completed	zachlewis	scanId = "1876", projectVersionId = "4287", notes = ""
7:09 AM	User Start Scan Static	zachlewis	scanId = "1876", notes = ""
6:46 AM	Release Created	zachlewis	ReleaseName = "Try2", SDLCStatus = "1", ReleaseRedirectUri = "https://sandbox.fortify.com/redirect/Releases/4287", ApplicationName = "Demo FPR Upload", Application "Web/Thick-Client", ApplicationBusinessCriticality = "1", ApplicationRedirectUri = "https://sandbox.fortify.com/redirect/Applications/4054"
6:32 AM	Static Scan Completed	zachlewis	scanId = "1875", projectVersionId = "4286", notes = ""
6:27 AM	User Start Scan Static	zachlewis	scanId = "1875", notes = ""
2:08 AM	Static Scan Rejected	zachlewis	notes = "Reason: Missing JAR/CLASS or JAVA files, Notes: Java payloads must contain .java, .jsp, .jar, or .class files"
2:03 AM	User Start Scan Static	zachlewis	scanId = "1874", notes = ""

FOD Reviewing Results

Understanding the Sink and what needs to be done.



FOD Reviewing Results - 2



FOD Reviewing Results - 3

70524 **basic/vbscript/sink.asp : 17**

Try2 | Critical | SQL Injection | SMART FIX

Vulnerability | Recommendations | Code | Diagram | More Evidence | History

Summary

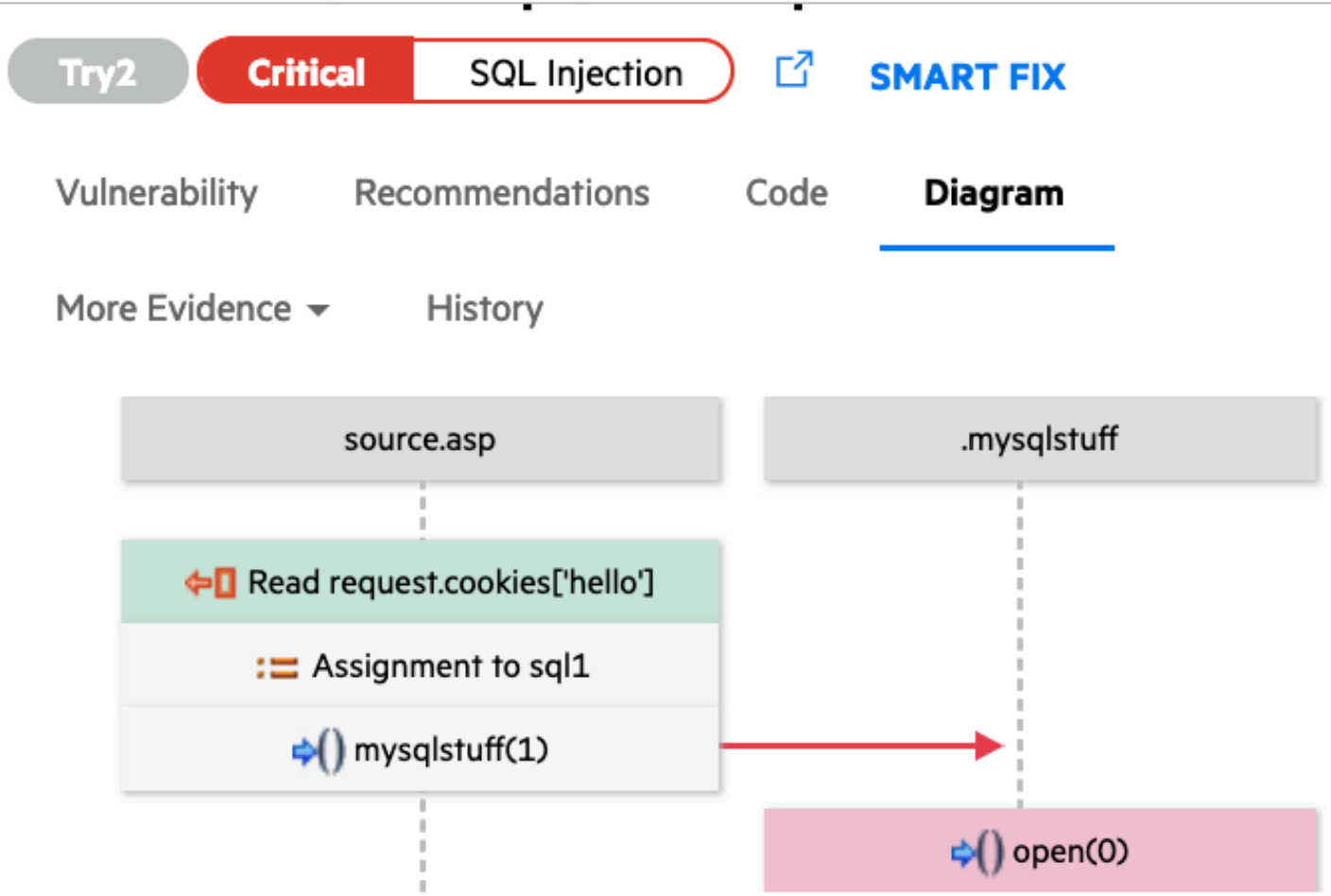
Line **17** of **sink.asp** invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.Constructing a dynamic SQL statement with input coming from an untrusted source could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

Explanation

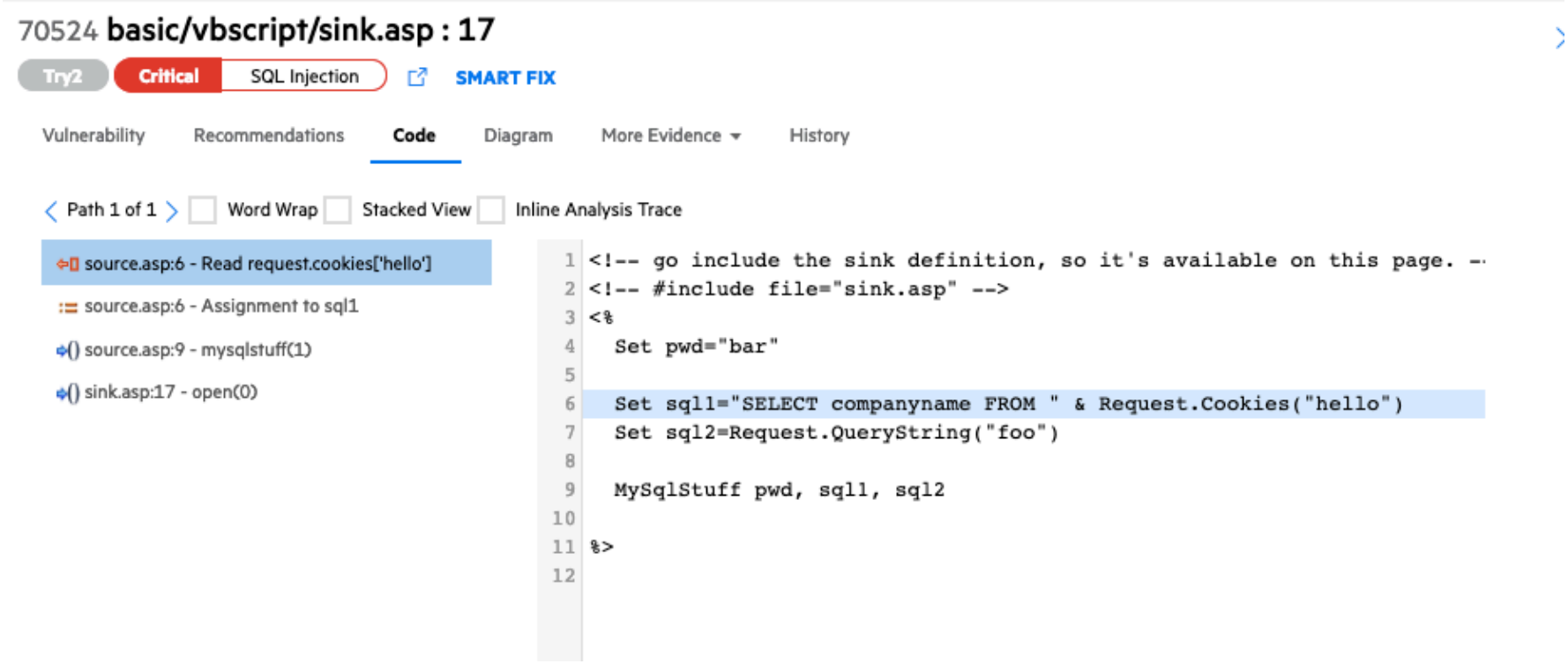
SQL injection errors occur when:

1. Data enters a program from an untrusted source.

FOD Reviewing Results - 4



FOD Reviewing Results - 5



FOD Reviewing Results - 6

Demo FPR Upload › Try2

Release Issues

70524 **basic/vbscript/sink.asp : 17**

Try2 | Critical | SQL Injection | SMART FIX

Vulnerability | **Recommendations** | Code | Diagram | More Evidence | History

Recommendation

The root cause of a SQL injection vulnerability is the ability of an attacker to change context in the SQL query, causing a value that the programmer intended to be interpreted as data to be interpreted as a command instead. When a SQL query is constructed, the programmer knows what should be interpreted as part of the command and what should be interpreted as data. Parameterized SQL statements can enforce this behavior by disallowing data-directed context changes and preventing nearly all SQL injection attacks. Parameterized SQL

the user to select from this set of application-controlled values.

Tips

1. A common mistake is to use parameterized SQL statements that are constructed by concatenating user-controlled strings. Of course, this defeats the purpose of using parameterized SQL statements. If you are not certain that the strings used to form parameterized statements are constants controlled by the application, do not assume that they are safe because they are not being executed directly as SQL strings. Thoroughly investigate all uses of user-controlled strings in SQL statements and verify that none can be used to modify the meaning of the query.

NEW Interactive Training

Through our partnership with Secure Code Warrior, try a short, hands-on challenge where sample software code needs to be analyzed for SQL Injection security weaknesses. Once identified, you can then modify the code to remediate or mitigate the SQL Injection vulnerability. Fortify on Demand does not share your personal or organization's information with Secure Code Warrior.

LAUNCH TRAINING

References

1. SQL Injection Attacks by Example, S. J. Friedl, http://www.unixwiz.net/techtips/sql-injection.html
2. Stop SQL Injection Attacks Before They Stop You, P. Litwin
3. SQL Injection and Oracle, Part One, P. Finnigan, http://www.securityfocus.com/infocus/1644
4. Writing Secure Code, Second Edition, M. Howard, D. LeBlanc
5. CWE ID 89, Standards Mapping - Common Weakness Enumeration

FOD Reviewing Results - 7

Common Weakness Enumeration: CWE

https://cwe.mitre.org/ ▼

Apr 3, 2018 - **Common Weakness Enumeration** (**CWE**) is a list of software weaknesses.

About CWE · CWE List · Search · CWE - Reports

CWE - CWE-89: Improper Neutralization of Special Elements used in ...

https://cwe.mitre.org › CWE List ▼

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ..... The programmer may have skipped any input validation on **$id** under the ...

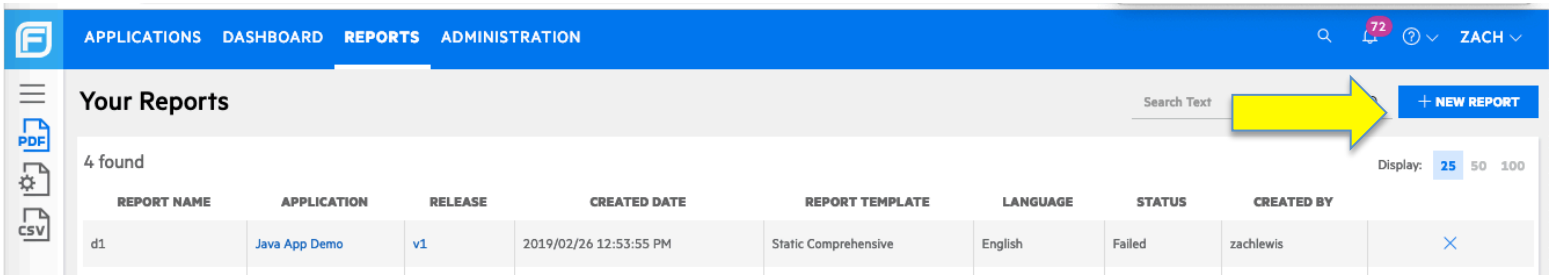
Phase: Implementation

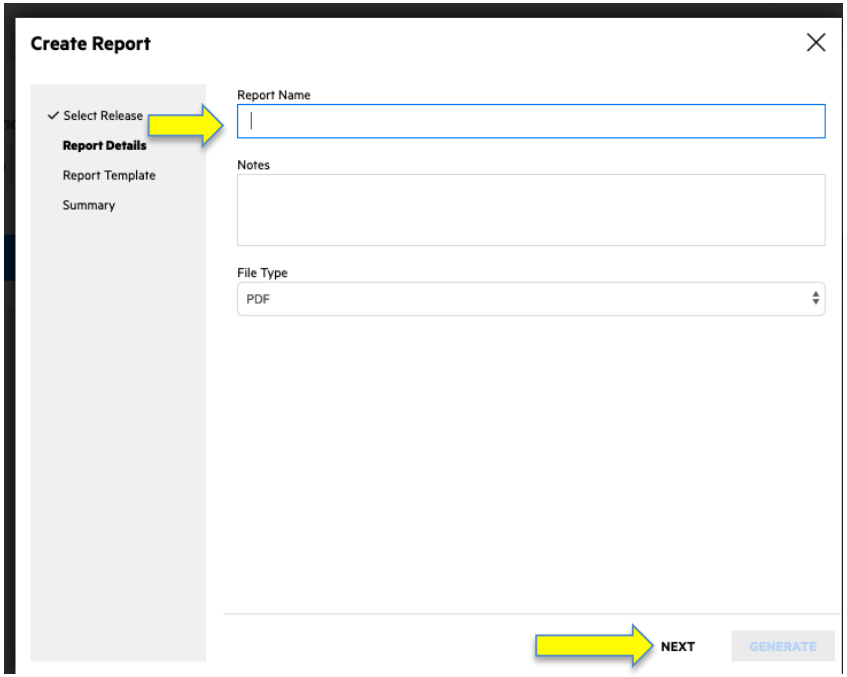
Strategy: Input Validation

Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does.

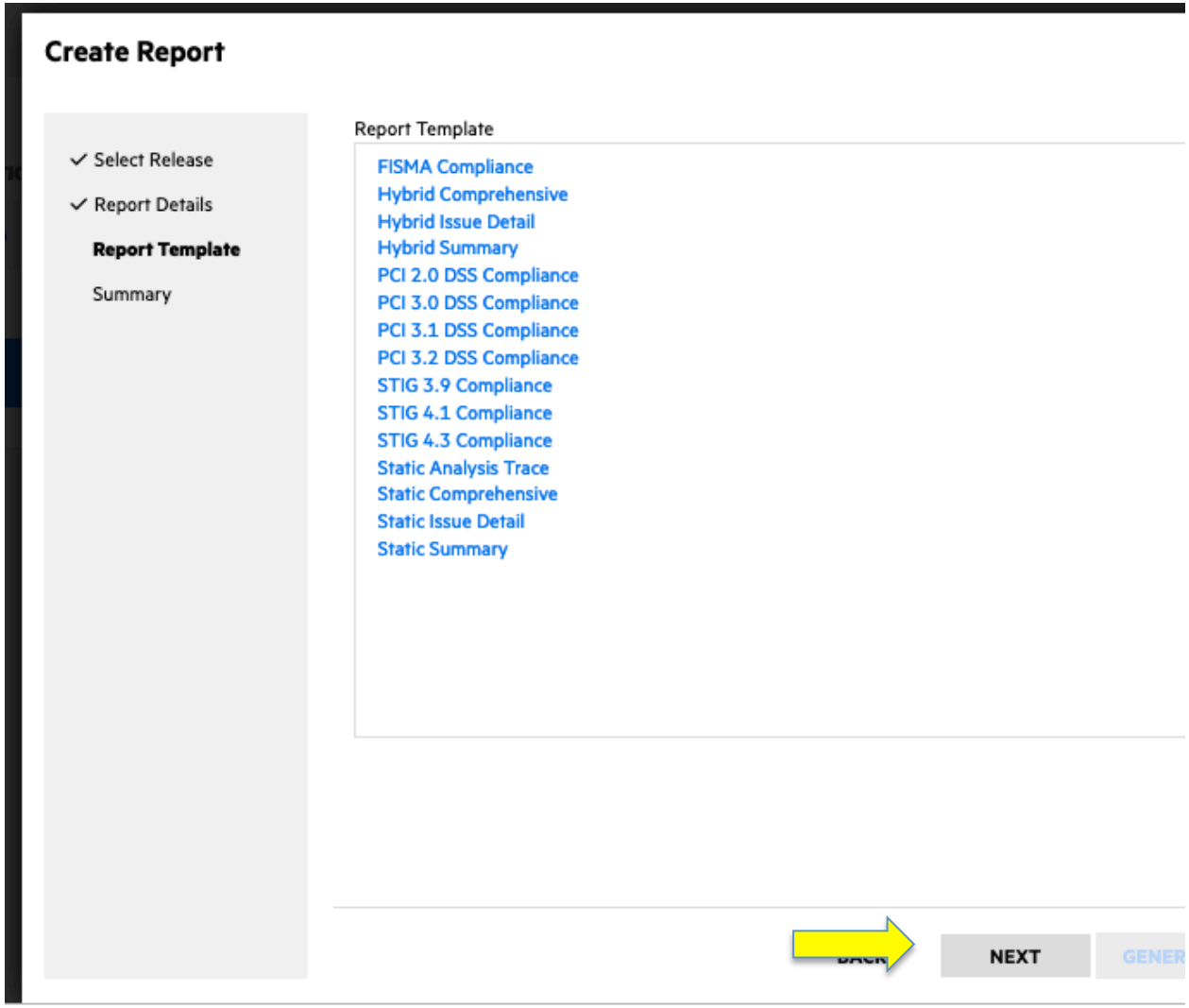
MemberOf	C	1005	7PK - Input Validation and Representation

FOD Reporting

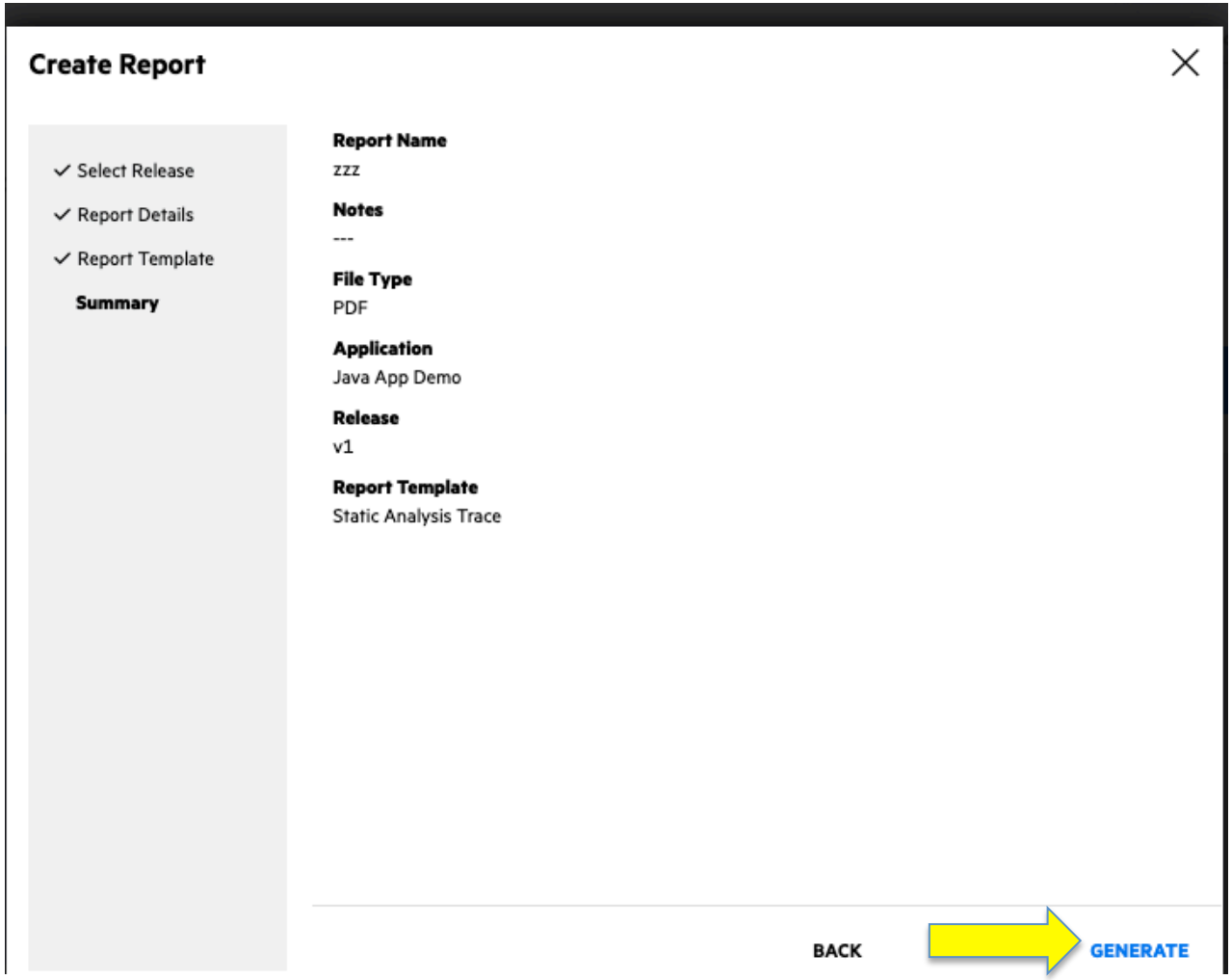




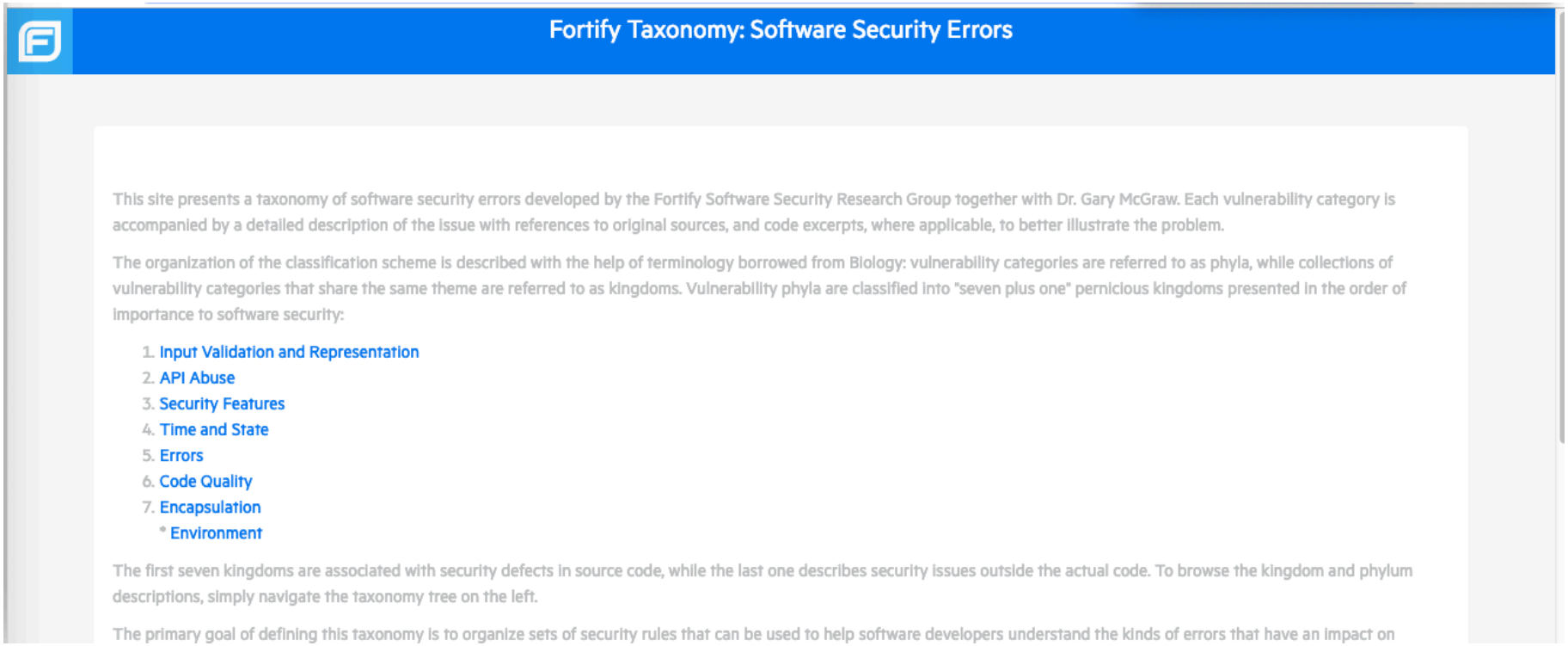
FOD Reporting - 2



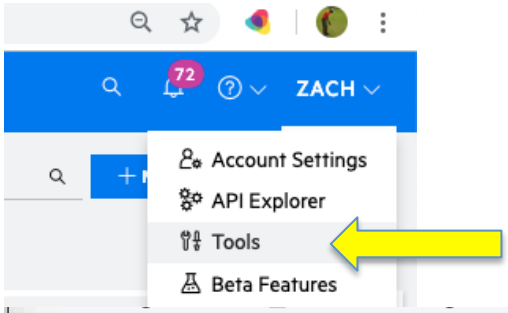
FOD Reporting - 3



https://vulncat.fortify.com/en

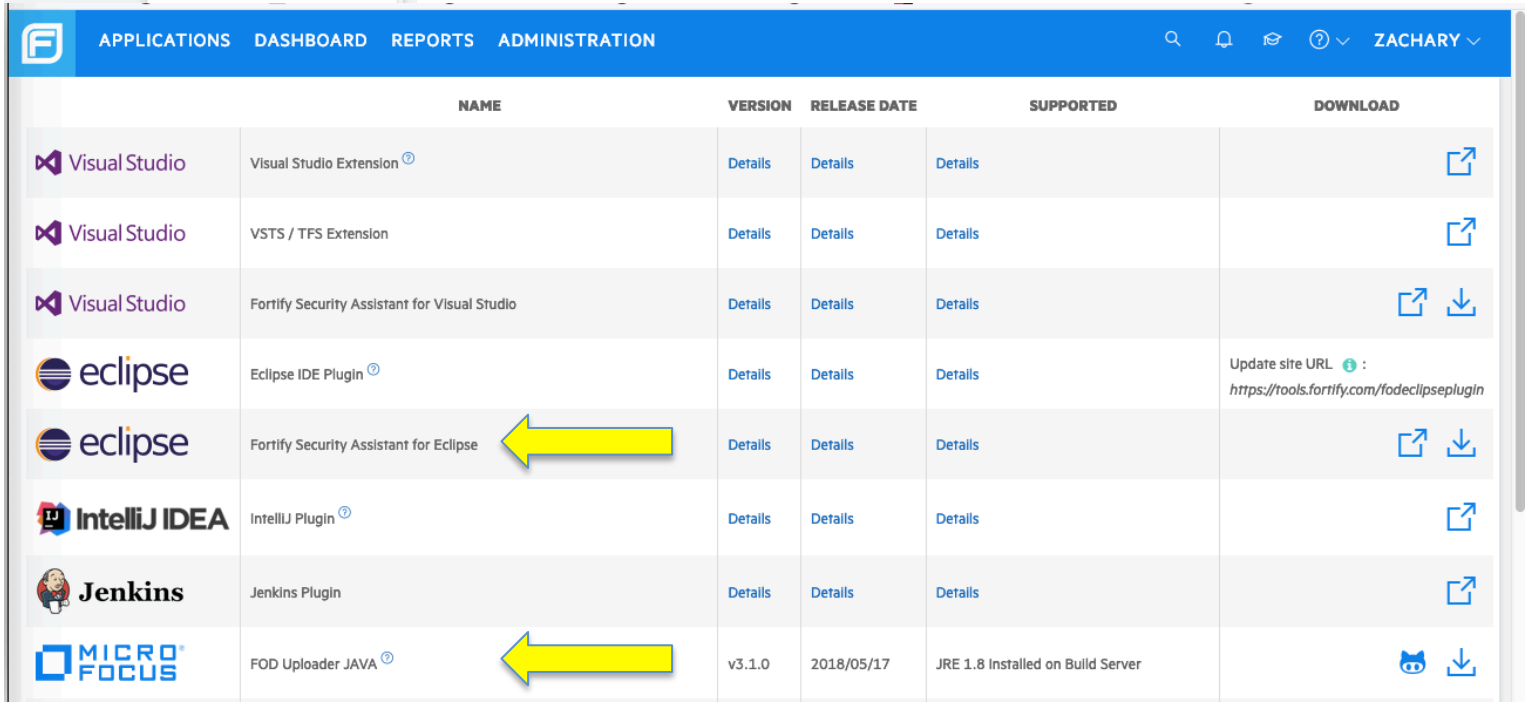


Key FOD Tools and Plugins



Select the Tools from Main Dashboard.

1. Eclipse Security Assistant
2. FODUploader.jar



Key FOD Tools and Plugins -2
FOD Uploader

jFortify on Demand provides a build server integration (BSI) tool called FoDUploader to help you upload applications from a build server. FoDUploader runs from the command-line .

Usagee:

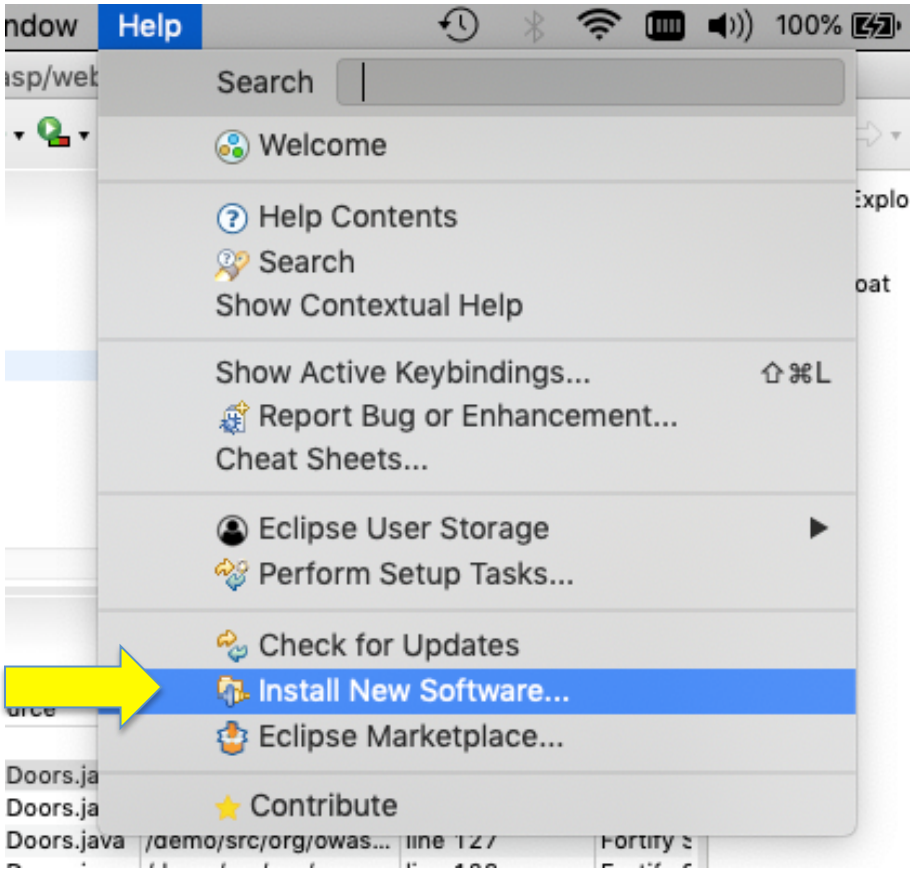
java -jar FodUpload.jar −apiCredentials <apikey> -zipLocation appsrc.zip -entitlementPreference 2 −bsi <bsi Key>

Example:

java -jar FodUpload.jar -apiCredentials "ddda12dc-39e1-4ab9-91b0-94ac4fa6065f" "eE5TYjR6PHZLNzNzRUNxZ0hhV2o6dzIicVBsSFc20" -zipLocation /basic.mbs.zip -entitlementPreference 2 -bsi eyJ0ZW5hbnRJZCI6MTIxOSwidGVuYW50Q29kZSI6ImZvcnRpZnlfemwiLCJyZWxlYXNlSWQiOjQyODUsInBheWxvYWRUeXBlIjoiQU5BTFlTSVNfUEFZTE9BRCIsImFzc2Vzc21lbnRUeXBlSWQiOjI1LCJ0ZWNobm9sb2d5VHlwZSI6Ik1CUy9DL0MrKy9TY2FsYSIsInRlY2hub2xvZ3lUeXBlSWQiOjE4LCJ0ZWNobm9sb2d5VmVyc2lvbiI6bnVsbCwidGVjaG5vbG9neVZlcnNpb25JZCI6bnVsbCwiYXVkaXRQcmVmZXJlbmNlIjoiQXV0b21hdGVkIiwiYXVkaXRQcmVmZXJlbmNlSWQiOjIsImluY2x1ZGVUaGlyZFBhcnR5IjpmYWxzZSwiaW5jbHVkZU9wZW5Tb3VyY2VBbmFseXNpcyI6ZmFsc2UsInBvcnRhbFVyaSI6Imh0dHBzOi8vc2FuZGJveC5mb3J0aWZ5LmNvbSIsImFwaVVyaSI6Imh0dHBzOi8vYXBpLnNhbmRib3guZm9ydGlmeS5jb20iLCJzY2FuUHJlZmVyZW5jZSI6IlN0YW5kYXJkIiwic2NhblByZWZlcmVuY2VJZCI6MX0=

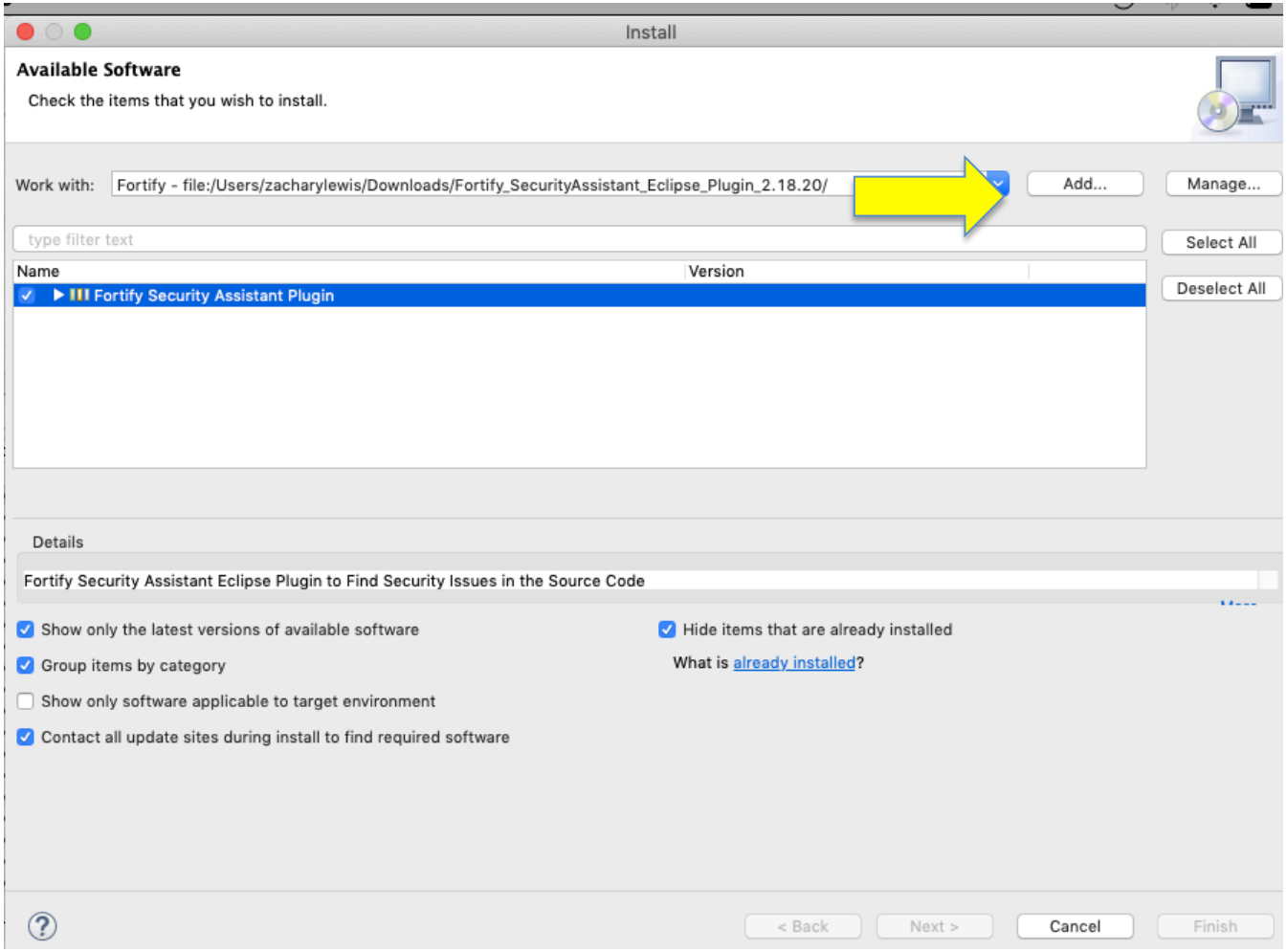
Authenticating
Beginning upload
Upload Status - Bytes sent:98147
Scan 1871 uploaded successfully. Total bytes sent: 98147
Retiring Token : Token Retired Successfully

Key FOD Tools and Plugins – 3
Installation of Eclipse Security Assistant Plugin



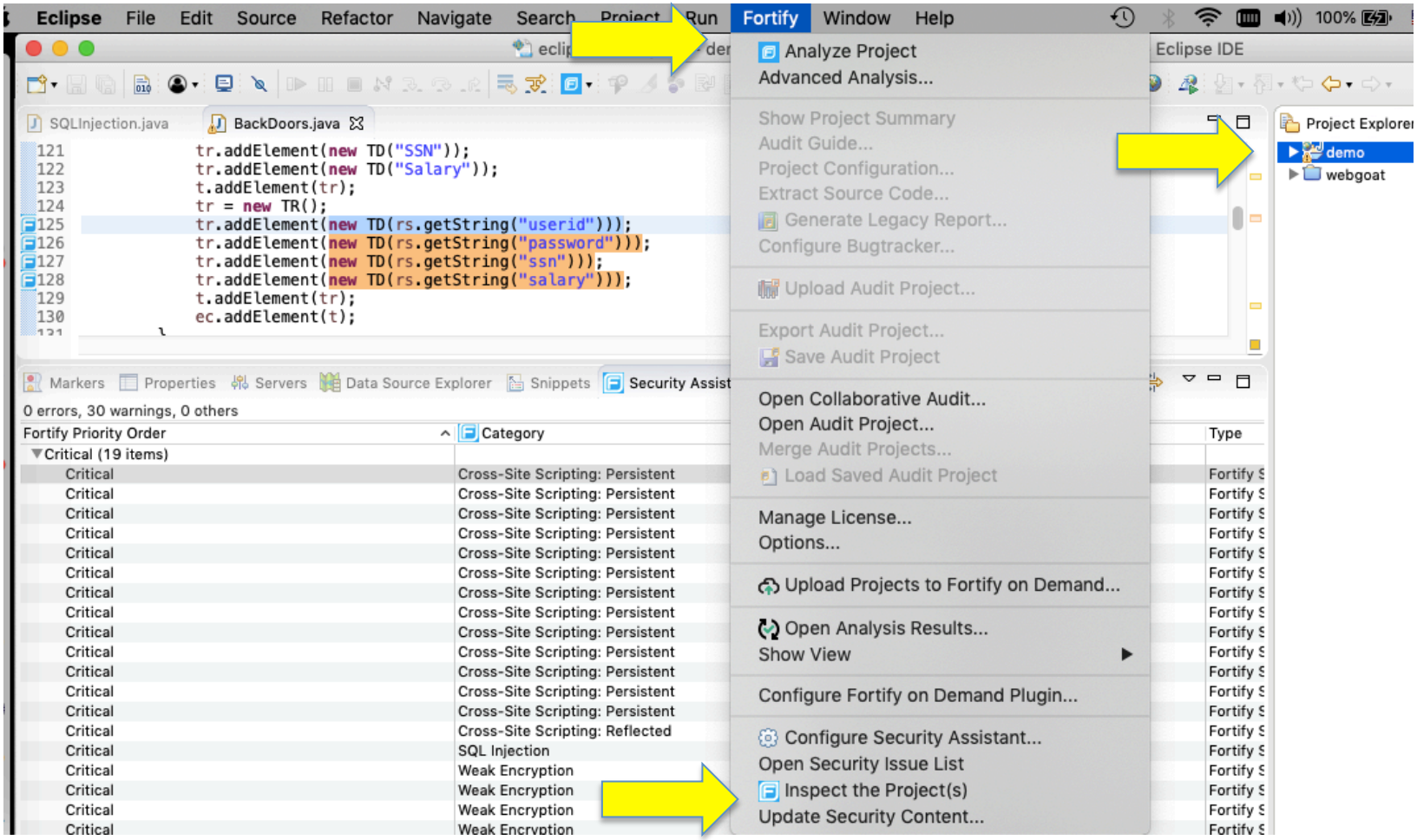
Key FOD Tools and Plugins – 4
Installation of Eclipse Security Assistant Plugin

Add Path to Downloaded unzipped FOD Eclipse Security Assistant



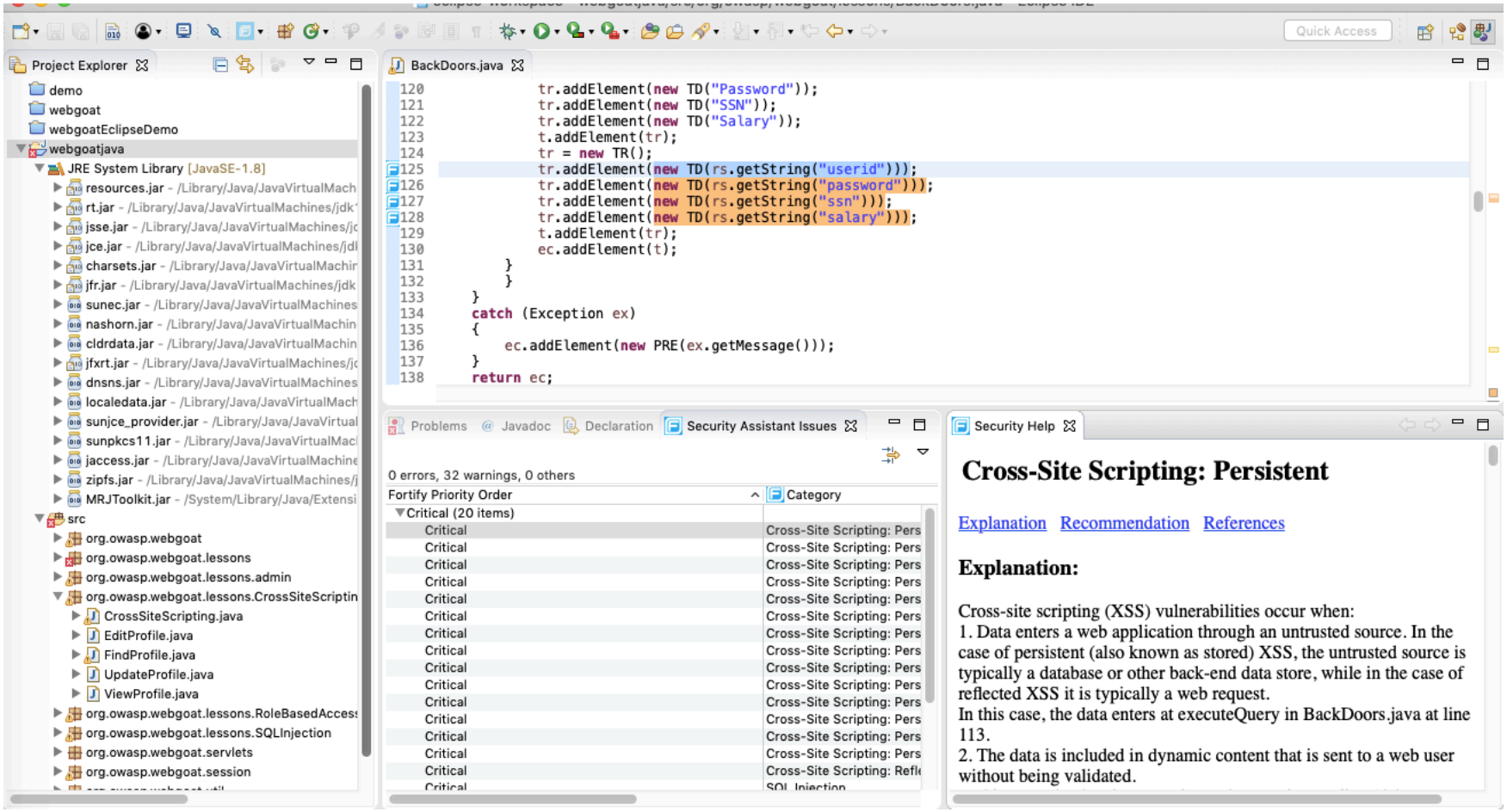
Key FOD Tools and Plugins - 5
Using Eclipse Security Assistant Plugin

NOTE: It must be a Java Project and selected to use Inspect Project



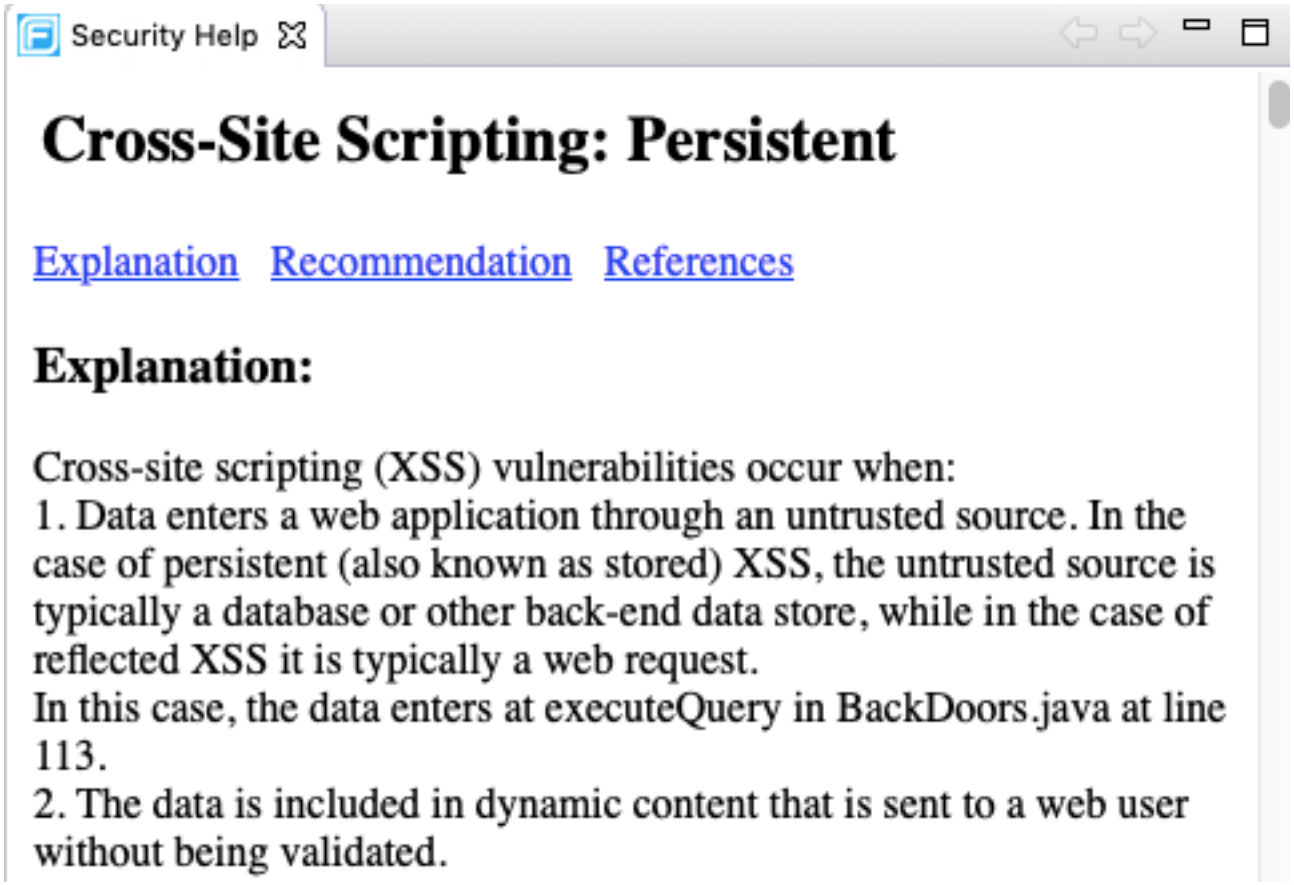
Key FOD Tools and Plugins - 6
Using Eclipse Security Assistant Plugin

NOTE: It must be a Java Project and selected to use Inspect Project

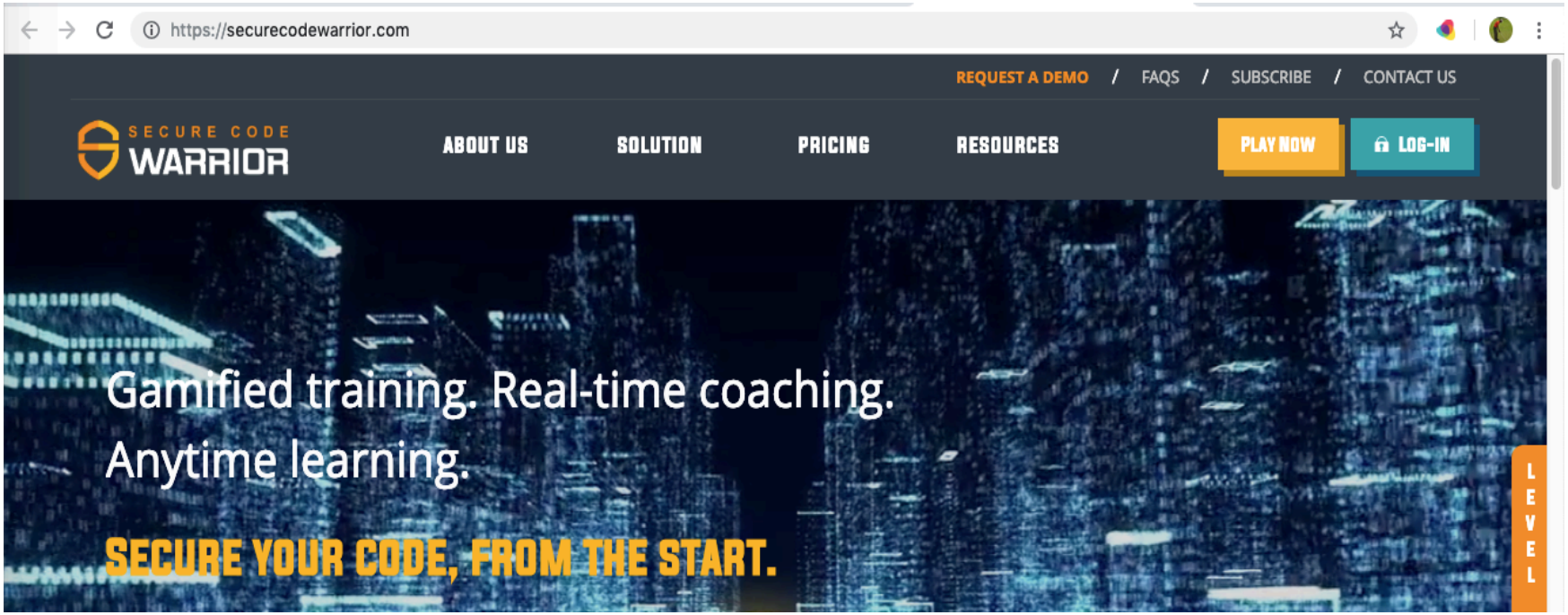


Key FOD Tools and Plugins - 7
Using Eclipse Security Assistant Plugin

NOTE: It must be a Java Project and selected to use Inspect Project



Code Warrior AppSec Training



Done