1
00:00:00,630 --> 00:00:03,390
Welcome to this presentation of shifting left

2
00:00:03,390 --> 00:00:05,760
the right way with OSCAL. This

3
00:00:05,760 --> 00:00:08,130
presentation has been prepared for you by

4
00:00:08,130 --> 00:00:12,150
Chris Compton, Alexander Stein, and Nikita Wootten.

5
00:00:12,810 --> 00:00:14,910
as a part of this presentation we

6
00:00:15,015 --> 00:00:16,425
will take a moment to touch on

7
00:00:16,383 --> 00:00:18,753
the overall OSCAL project for anyone

8
00:00:18,690 --> 00:00:19,791
new to the project.

9
00:00:20,550 --> 00:00:22,320
We will then talk about a workflow

10
00:00:22,350 --> 00:00:23,580
we have used as a part of

11
00:00:23,580 --> 00:00:26,430
a research project at NIST as well

12
00:00:26,430 --> 00:00:28,710
as share a demonstration of the approach.

13
00:00:30,750 --> 00:00:34,710

This started as a research pilot for

14
00:00:34,740 --> 00:00:38,640
secure information exchange between multiple organizations that

15
00:00:38,640 --> 00:00:42,120
is continuously monitored assessed and authorised to

16
00:00:42,120 --> 00:00:42,750
operate.

17
00:00:45,690 --> 00:00:49,050
Basically when you embark on a project

18
00:00:49,050 --> 00:00:52,920
like this you have a control library

19
00:00:53,250 --> 00:00:55,200
that you're using as the basis for

20
00:00:55,200 --> 00:00:58,440
your controls you have a process to

21
00:00:58,440 --> 00:01:01,350
apply that and then you have this

22
00:01:01,350 --> 00:01:03,660
black box that is

23
00:01:04,680 --> 00:01:08,070
kind of a just a broad array

24
00:01:08,070 --> 00:01:11,730
of documentation and output from the work

25
00:01:11,730 --> 00:01:14,040
that is performed through the process.

26
00:01:16,142 --> 00:01:20,342
OSCAL was created as the glue in

27
00:01:20,330 --> 00:01:21,780
this black box

28
00:01:22,890 --> 00:01:26,370
to have a common specification for producing

29
00:01:26,370 --> 00:01:30,300
documentation around this process.

30
00:01:31,620 --> 00:01:33,840
The goal of OSCAL is to produce

31
00:01:33,870 --> 00:01:37,860
and interpret machine readable security documentation using

32
00:01:37,890 --> 00:01:42,270
a common specification that promotes interoperability. So

33
00:01:42,270 --> 00:01:44,070
on the right side of the slide

34
00:01:44,070 --> 00:01:47,100
in very small print you see what

35
00:01:47,100 --> 00:01:50,610
would be the traditional hundreds of pages

36
00:01:50,610 --> 00:01:51,600
of documentation

37
00:01:51,630 --> 00:01:54,690
and that is produced in word documents

38
00:01:54,750 --> 00:01:58,680

and spreadsheets and there are a number

39
00:01:58,680 --> 00:02:02,640
of issues with sharing that information between

40
00:02:03,090 --> 00:02:06,360
assessors and authorizing officials.

41
00:02:09,000 --> 00:02:10,710
So this is kind of an overview

42
00:02:10,710 --> 00:02:13,020
of what OSCAL allows you to do

43
00:02:13,260 --> 00:02:15,720
and that is to take a wide

44
00:02:15,720 --> 00:02:19,710
array of documents spreadsheets proprietary tools

45
00:02:21,270 --> 00:02:25,650
custom tools and using OSCAL as

46
00:02:25,650 --> 00:02:28,770
a common data specification you can create

47
00:02:29,220 --> 00:02:33,360
documents in xml json and yaml

48
00:02:34,950 --> 00:02:39,182
and within tools that are created in

49
00:02:39,182 --> 00:02:41,460
the the grc tools and services

50
00:02:42,870 --> 00:02:47,130
they are able to ingest these documents

51
00:02:47,190 --> 00:02:49,800
and produce these documents as vendors add

52
00:02:49,800 --> 00:02:54,480
this functionality to the products and so

53
00:02:54,480 --> 00:02:58,140
it facilitates sharing this information across different

54
00:02:58,170 --> 00:03:00,930
types of software it also gives you

55
00:03:00,930 --> 00:03:02,640
the ability to create

56
00:03:02,640 --> 00:03:05,970
documents from these machine readable formats to

57
00:03:05,970 --> 00:03:08,820
produce the original word documents if you

58
00:03:08,820 --> 00:03:12,360
needed those or spreadsheets and those kinds

59
00:03:12,360 --> 00:03:13,020
of things.

60
00:03:16,650 --> 00:03:21,840
This enables also automated traceability from selection

61
00:03:21,840 --> 00:03:26,670
of security controls through implementation and assessment

62
00:03:26,730 --> 00:03:28,170
and i'll show that in the next

63
00:03:28,170 --> 00:03:28,770

slide

64
00:03:32,160 --> 00:03:33,960
which gives us an overview of the

65
00:03:33,960 --> 00:03:36,300
models that are available in OSCAL.

66
00:03:39,420 --> 00:03:41,190
A number of graphs will show the

67
00:03:41,190 --> 00:03:43,710
arrows going in a reverse direction which

68
00:03:43,710 --> 00:03:45,840
are really focused on the traceability of

69
00:03:45,840 --> 00:03:47,580
the documents but this one is kind

70
00:03:47,580 --> 00:03:49,050
of to give you the impression of

71
00:03:49,050 --> 00:03:53,280
how information flows from the controls layer

72
00:03:53,790 --> 00:03:55,890
all the way through implementation in the

73
00:03:55,890 --> 00:03:57,570
assessment layer so you would

74
00:03:57,930 --> 00:04:01,230
have a catalog that may have you

75
00:04:01,230 --> 00:04:03,660
know twelve hundred controls in it you

76
00:04:03,660 --> 00:04:07,530
can use a profile that allows you

77
00:04:07,530 --> 00:04:11,220
to subset the catalog so to speak

78
00:04:11,250 --> 00:04:14,430
into specifically what you're using for your

79
00:04:14,910 --> 00:04:17,760
implementation. That profile

80
00:04:17,760 --> 00:04:20,430
allows you to transform a new

81
00:04:20,430 --> 00:04:22,920
catalog with only those controls.

82
00:04:24,240 --> 00:04:28,290
That catalogue is imported into this system

83
00:04:28,290 --> 00:04:29,430
security plan.

84
00:04:30,570 --> 00:04:36,660
You're able to leverage other security plans

85
00:04:37,140 --> 00:04:42,420
for leverage authorizations to inherit of layers

86
00:04:43,050 --> 00:04:47,670
you can also build components that are

87
00:04:47,670 --> 00:04:49,920
somewhat like legos where you can pull

88
00:04:49,920 --> 00:04:50,640

those into

89
00:04:50,670 --> 00:04:54,300
your security plan for individual components of

90
00:04:54,300 --> 00:04:56,190
a system that may be reused

91
00:04:57,540 --> 00:05:00,330
and then that flows over into your

92
00:05:00,390 --> 00:05:03,540
assessment plans the results that can be

93
00:05:03,540 --> 00:05:05,610
produced off of that plan as well

94
00:05:05,610 --> 00:05:08,400
as the plan of action and milestones.

95
00:05:08,970 --> 00:05:11,040
We also have models that are in

96
00:05:11,040 --> 00:05:14,340
development related to control mapping and ones

97
00:05:14,340 --> 00:05:16,590
that we're looking at related to shared

98
00:05:16,590 --> 00:05:17,520
responsibilities.

99
00:05:22,680 --> 00:05:25,620
On the NIST website we also have a

100
00:05:25,620 --> 00:05:27,870
list of tools there there are a

101
00:05:27,870 --> 00:05:29,340
few tools that we make as a

102
00:05:29,340 --> 00:05:31,830
part of NIST one of those is

103
00:05:31,830 --> 00:05:34,320
our command line interface that allows you

104
00:05:34,320 --> 00:05:37,110
to translate between the different formats as

105
00:05:37,110 --> 00:05:38,880
well as perform validation

106
00:05:38,880 --> 00:05:39,720
of the models.

107
00:05:41,250 --> 00:05:44,640
We also have a list of open

108
00:05:44,640 --> 00:05:47,070
source and commercial license tools that are

109
00:05:47,070 --> 00:05:47,850
available.

110
00:05:49,140 --> 00:05:51,090
And even though we have those listed

111
00:05:51,090 --> 00:05:54,930
that does not imply endorsement or recommendation

112
00:05:54,930 --> 00:05:55,980
by NIST.

113
00:05:58,410 --> 00:06:00,420

We encourage you to give OSCAL

114
00:06:00,420 --> 00:06:02,460
a try if you have not you

115
00:06:02,460 --> 00:06:05,310
can also get involved with the project

116
00:06:05,310 --> 00:06:08,130
itself on the website we have a

117
00:06:08,130 --> 00:06:10,650
number of past workshops we have the

118
00:06:10,650 --> 00:06:14,340
reference documentation for OSCAL we

119
00:06:14,340 --> 00:06:17,280
have an updated set of community teleconference

120
00:06:17,280 --> 00:06:18,420
does that are available

121
00:06:19,590 --> 00:06:23,310
and we encourage contributions of code expertise

122
00:06:23,370 --> 00:06:24,780
and experiences.

123
00:06:26,280 --> 00:06:28,440
And now we'll have a discussion of

124
00:06:28,440 --> 00:06:30,270
the workflow around OSCAL.

125
00:06:32,220 --> 00:06:35,340
So let's talk about workflow

126
00:06:36,600 --> 00:06:39,660
how you integrate oscar content into a

127
00:06:39,660 --> 00:06:42,540
new or existing development or operations program

128
00:06:42,600 --> 00:06:45,750
could be varied and very different but

129
00:06:45,750 --> 00:06:47,370
we're going to show you a representative

130
00:06:47,370 --> 00:06:48,060
example

131
00:06:49,680 --> 00:06:51,300
that we made and hopefully that can

132
00:06:51,300 --> 00:06:54,270
speak to you and your development operations

133
00:06:54,270 --> 00:06:57,510
or other role and how you would

134
00:06:57,510 --> 00:07:00,180
use aws how your own systems in

135
00:07:00,180 --> 00:07:03,360
your own designs so for example we're

136
00:07:03,360 --> 00:07:05,040
going to focus on a devops approach

137
00:07:05,040 --> 00:07:07,650
in a certain technology stack for our

138
00:07:07,650 --> 00:07:09,360

conversation in this presentation.

139
00:07:09,360 --> 00:07:12,510
These choices make us comfortable and

140
00:07:12,510 --> 00:07:14,820
are either free or low-cost they are

141
00:07:14,820 --> 00:07:16,680
good for demonstrating how one can operate

142
00:07:16,680 --> 00:07:18,270
but it's not meant to be authoritative

143
00:07:18,270 --> 00:07:22,350
or exhausted on the topic of workflows

144
00:07:22,350 --> 00:07:26,670
for integrating OSCAL security documentation into

145
00:07:26,670 --> 00:07:29,340
your processes or your procedures or operations.

146
00:07:30,060 --> 00:07:31,680
They're good for demonstrating how one

147
00:07:31,680 --> 00:07:35,220
can operate but it's really to show

148
00:07:35,250 --> 00:07:37,440
through an experiment you can act like

149
00:07:37,440 --> 00:07:38,820
us or with us but without having

150
00:07:38,820 --> 00:07:40,320
to use your own work or technical

151
00:07:40,320 --> 00:07:42,690
stack it reduces barriers to sharing and

152
00:07:42,690 --> 00:07:45,000
practicing OSCAL without the high risk

153
00:07:45,000 --> 00:07:47,880
of doing so immediately in your production

154
00:07:47,880 --> 00:07:49,170
day to day work if you're doing

155
00:07:49,170 --> 00:07:49,530
this

156
00:07:49,680 --> 00:07:51,000
in a professional capacity.

157
00:07:52,200 --> 00:07:53,610
You don't have to have a combined

158
00:07:53,610 --> 00:07:55,830
team that handles development and operations together

159
00:07:55,830 --> 00:07:57,510
but we like that and that's how

160
00:07:57,510 --> 00:07:59,310
we're going to represent this OSCAL is

161
00:07:59,310 --> 00:08:00,960
not pedantic about that you do not

162
00:08:00,960 --> 00:08:02,760
have to use OSCAL specifically that way.

163
00:08:03,390 --> 00:08:05,250

We have a team to develop and

164
00:08:05,280 --> 00:08:08,520
operate an example application with python and

165
00:08:08,520 --> 00:08:10,530
using shell scripts in presumably a linux

166
00:08:10,530 --> 00:08:12,240
environment we will show

167
00:08:12,240 --> 00:08:13,500
show you how to use a workflow

168
00:08:13,500 --> 00:08:15,660
with GitHub actions to take work from

169
00:08:15,660 --> 00:08:17,580
our developers build it test it and

170
00:08:17,580 --> 00:08:19,440
potentially deploy it. This does not mean

171
00:08:19,440 --> 00:08:20,850
that you have to use this and

172
00:08:20,850 --> 00:08:22,320
only this for OSCAL you can

173
00:08:22,320 --> 00:08:25,260
use a variety of integration environments and

174
00:08:25,260 --> 00:08:28,380
workflows manual semi automated or fully automatic.

175
00:08:28,440 --> 00:08:29,940
This is just what makes us comfortable

176
00:08:29,940 --> 00:08:31,530
and we're hoping it will speak with

177
00:08:31,530 --> 00:08:32,400
us to you

178
00:08:32,400 --> 00:08:34,110
oh and you can always ask us

179
00:08:34,110 --> 00:08:36,510
questions about other ways you can use

180
00:08:36,510 --> 00:08:38,159
it in your own workflows OSCAL is

181
00:08:38,159 --> 00:08:40,740
flexible enough and interoperable enough with a

182
00:08:40,740 --> 00:08:43,919
variety of programming languages runtimes operating systems

183
00:08:43,950 --> 00:08:46,530
and different kinds of organizational structure.

184
00:08:46,530 --> 00:08:49,020
So on that note you might ask why

185
00:08:49,020 --> 00:08:50,310
we would even do this and let's

186
00:08:50,310 --> 00:08:51,660
go return to the title of his

187
00:08:51,660 --> 00:08:52,320
presentation.

188
00:08:52,800 --> 00:08:54,690

An often heard critique of security design

189
00:08:54,720 --> 00:08:57,600
implementation and operation in many IT projects

190
00:08:57,690 --> 00:09:00,060
whether it's using somebody else's tools and

191
00:09:00,120 --> 00:09:01,710
deploying in your environment or writing your

192
00:09:01,710 --> 00:09:03,780
own things and deploying in your environment

193
00:09:04,470 --> 00:09:06,060
security is often too little too late.

194
00:09:06,690 --> 00:09:09,390
Many practitioners not just nist believe that

195
00:09:09,390 --> 00:09:11,910
the cost of securing projects in software

196
00:09:11,910 --> 00:09:12,720
and systems is all

197
00:09:12,720 --> 00:09:14,640
always been cheaper when is that earlier

198
00:09:14,640 --> 00:09:16,800
in the life cycle. You've probably heard

199
00:09:16,800 --> 00:09:18,900
people describe this as shifting left. Increasing

200
00:09:18,900 --> 00:09:22,350
secure design implementation and operation as early

201
00:09:22,350 --> 00:09:23,640
in the life cycle for a system as

202
00:09:23,640 --> 00:09:25,440
possible to account for it being cheaper

203
00:09:25,440 --> 00:09:26,430
in the first steps of the life

204
00:09:26,430 --> 00:09:29,160
cycle when planning and designing system not

205
00:09:29,160 --> 00:09:30,397
later when many of the things are

206
00:09:30,397 --> 00:09:32,880
complemented or operational in every state

207
00:09:32,880 --> 00:09:34,650
each of the life cycle increase security

208
00:09:34,650 --> 00:09:37,350
requires increased understanding and that means documenting

209
00:09:37,350 --> 00:09:40,170
and explaining the design the implementation operation

210
00:09:40,200 --> 00:09:41,730
not to the people that just develop

211
00:09:41,740 --> 00:09:44,160
and operate it but other stakeholders and

212
00:09:44,160 --> 00:09:46,650
other people in the development and operations

213
00:09:46,650 --> 00:09:49,470

teams. So how does OSCAL help

214
00:09:49,470 --> 00:09:51,990
document those things and if OSCAL can

215
00:09:51,990 --> 00:09:53,010
help document those things

216
00:09:53,040 --> 00:09:54,360
we hope that it does and will

217
00:09:54,360 --> 00:09:56,460
show you in this example how can

218
00:09:56,460 --> 00:09:58,170
OSCAL document the security

219
00:09:58,170 --> 00:10:00,360
requirements and how they're met the first

220
00:10:00,360 --> 00:10:02,460
stages of the life cycle for a

221
00:10:02,460 --> 00:10:05,310
system and facilitate a feedback loop between

222
00:10:05,310 --> 00:10:07,980
project managers and developers in a way

223
00:10:07,980 --> 00:10:09,990
that increases momentum in the life cycle

224
00:10:10,320 --> 00:10:12,060
to make sure that the document understanding

225
00:10:12,060 --> 00:10:13,200
keeps pace with the changes

226
00:10:13,625 --> 00:10:15,275
as they happen

227
00:10:19,115 --> 00:10:20,285
and how we'll do that is with

228
00:10:20,285 --> 00:10:22,235
this example we were talking before which

229
00:10:22,235 --> 00:10:24,425
we represented with this high level diagram

230
00:10:24,425 --> 00:10:26,855
of the workflow. So we were talking

231
00:10:26,855 --> 00:10:28,505
before about the kind of abstract ten

232
00:10:28,505 --> 00:10:31,625
thousand foot view this is a visualization

233
00:10:31,625 --> 00:10:34,325
of the different components of our automated

234
00:10:34,325 --> 00:10:36,665
workflow and how it was implemented using

235
00:10:36,665 --> 00:10:38,555
services provided by GitHub

236
00:10:39,815 --> 00:10:41,405
when we use GitHub actions that allows

237
00:10:41,405 --> 00:10:44,255
project managers and developers and other members

238
00:10:44,255 --> 00:10:45,965

of the development or operations team to

239
00:10:45,965 --> 00:10:48,515
add canonical copies or references to other

240
00:10:48,515 --> 00:10:51,875
documentation plans procedures and keep it close

241
00:10:51,875 --> 00:10:54,125
to the code itself and that code

242
00:10:54,815 --> 00:10:57,035
will help build and test the system

243
00:10:57,155 --> 00:10:59,957
and keep it close as possible to the system

244
00:10:59,945 --> 00:11:01,955
that is being developed and documented

245
00:11:01,955 --> 00:11:04,625
within it. This allows not just developers but

246
00:11:04,625 --> 00:11:06,695
other non developer staff to benefit from

247
00:11:06,695 --> 00:11:09,905
structured content and get just-in-time feedback not

248
00:11:09,905 --> 00:11:11,225
just for the code but the security

249
00:11:11,225 --> 00:11:13,865
documentation for this understanding of the system

250
00:11:13,865 --> 00:11:16,145
that's very necessary for different pieces of

251
00:11:16,145 --> 00:11:18,665
OSCAL content. So this is a huge

252
00:11:18,665 --> 00:11:20,105
time-saver and boost

253
00:11:20,105 --> 00:11:22,145
where many forms of security documentation

254
00:11:22,745 --> 00:11:25,385
without OSCAL required manual review by informed

255
00:11:25,385 --> 00:11:26,855
and skilled staff that takes money and

256
00:11:26,855 --> 00:11:29,405
time. Automating some of these pieces is

257
00:11:29,405 --> 00:11:30,935
demonstrated as the parts that will show

258
00:11:30,935 --> 00:11:32,285
here the diagram and later in a

259
00:11:32,285 --> 00:11:35,734
presentation and demo that saves time for

260
00:11:35,885 --> 00:11:37,625
expert staff that need to be focusing

261
00:11:37,625 --> 00:11:40,265
their efforts elsewhere. So as we show

262
00:11:40,355 --> 00:11:43,835
here in the diagram alongside application testing

263
00:11:43,835 --> 00:11:46,265

that happens in GitHub we also allow

264
00:11:46,265 --> 00:11:48,035
people to add OSCAL contents of the

265
00:11:48,065 --> 00:11:51,545
repo and the workflow implemented by code

266
00:11:51,545 --> 00:11:53,375
this defined in GitHub actions follows a

267
00:11:53,375 --> 00:11:56,225
series of steps consistently for every piece

268
00:11:56,225 --> 00:11:58,295
of chunk of code changed or documentation

269
00:11:58,295 --> 00:12:00,425
updated to the system in

270
00:12:00,386 --> 00:12:01,946
this repository for the code in GitHub.

271
00:12:02,021 --> 00:12:04,871
The validation action as you can

272
00:12:04,835 --> 00:12:07,505
see in the second column looks at

273
00:12:07,505 --> 00:12:09,005
the OSCAL content that's present in the

274
00:12:09,005 --> 00:12:11,915
repository and checks it's structure for each

275
00:12:11,915 --> 00:12:13,475
of the document types and does logic

276
00:12:13,475 --> 00:12:18,035
checking for common mistakes. So for the

277
00:12:18,035 --> 00:12:20,585
security requirements that are particular to this

278
00:12:20,585 --> 00:12:22,655
system whether they're planned or implemented those

279
00:12:22,655 --> 00:12:25,535
can be defined in a profile. Those

280
00:12:25,535 --> 00:12:27,845
are referencing and must cross reference and

281
00:12:27,845 --> 00:12:29,825
check what we'll call a database of

282
00:12:29,825 --> 00:12:32,555
all the collective controls that can be

283
00:12:32,645 --> 00:12:34,385
pertaining to a system from some group be it

284
00:12:34,385 --> 00:12:36,545
at NIST or elsewhere and that kind

285
00:12:36,545 --> 00:12:38,585
of thing that's like a database is

286
00:12:38,585 --> 00:12:40,715
filtered down is a catalog

287
00:12:40,745 --> 00:12:42,365
that also was checked if it is

288
00:12:42,365 --> 00:12:44,705

either in the repository or cross referenced

289
00:12:44,795 --> 00:12:46,085
in some remote location.

290
00:12:47,255 --> 00:12:49,715
The actual requirements as planned or implemented

291
00:12:49,715 --> 00:12:51,305
for the system that we're developing

292
00:12:52,535 --> 00:12:54,935
are defined in a system security plan

293
00:12:55,175 --> 00:12:57,995
and that is also checked alongside an

294
00:12:57,995 --> 00:13:00,755
assessment plan that will be

295
00:13:00,755 --> 00:13:02,465
used in a subsequent column that defines

296
00:13:02,465 --> 00:13:04,685
the tests automated or not but we'll

297
00:13:04,685 --> 00:13:07,565
focus on automated tests that can take

298
00:13:07,565 --> 00:13:10,325
the information about planned and implemented security

299
00:13:10,325 --> 00:13:13,761
requirements in the system in the system security plan

300
00:13:13,772 --> 00:13:16,475
and test those for suitability.

301
00:13:18,125 --> 00:13:20,825
Once that is done GitHub actions

302
00:13:20,825 --> 00:13:22,355
can be used to actually read the

303
00:13:22,355 --> 00:13:24,605
assessment plan which is in the repository

304
00:13:24,665 --> 00:13:26,315
and through accustomed action is defined in

305
00:13:26,315 --> 00:13:28,565
the third column read the assessment plan

306
00:13:28,565 --> 00:13:31,805
to actually execute automated testing and those

307
00:13:31,805 --> 00:13:35,735
tests define how certain security requirements are

308
00:13:35,735 --> 00:13:37,865
met from the controls that are defined

309
00:13:37,865 --> 00:13:40,355
by NIST or ISO or other

310
00:13:40,355 --> 00:13:42,545
groups and what the state of them

311
00:13:42,845 --> 00:13:43,805
are in the system.

312
00:13:44,825 --> 00:13:46,475
And then they will generate a result

313
00:13:46,535 --> 00:13:48,275

of an assessment and that is an

314
00:13:48,425 --> 00:13:50,225
OSCAL called in assessment result.

315
00:13:51,305 --> 00:13:52,865
In the final stage in the fourth

316
00:13:52,865 --> 00:13:54,095
column you can see that there is

317
00:13:54,095 --> 00:13:57,215
custom action that will validate the assessment

318
00:13:57,215 --> 00:14:01,085
results which includes all the tests as

319
00:14:01,085 --> 00:14:03,425
executed in the plan and after that

320
00:14:03,425 --> 00:14:05,915
validated for the benefit of developers and

321
00:14:05,915 --> 00:14:09,485
non-developers alike the details of the different

322
00:14:09,485 --> 00:14:11,465
results will be uploaded

323
00:14:11,465 --> 00:14:13,565
and presented in the web interface

324
00:14:13,565 --> 00:14:16,025
of the GitHub system as code

325
00:14:16,025 --> 00:14:18,245
changes occur in GitHub we call this

326
00:14:18,245 --> 00:14:19,835
a pull request. So this allows both

327
00:14:19,835 --> 00:14:23,315
developers and non-developers to actually see the

328
00:14:23,315 --> 00:14:25,805
results of testing as system changes are

329
00:14:25,805 --> 00:14:27,755
being made in the code that defines

330
00:14:27,815 --> 00:14:29,855
how the system will be operated and

331
00:14:29,855 --> 00:14:33,191
deployed as they are made before they

332
00:14:33,215 --> 00:14:35,383
are finally reviewed and approved by the team.

333
00:14:40,355 --> 00:14:41,405
And of course we were talking about

334
00:14:41,405 --> 00:14:43,385
what the security requirements looked like here's

335
00:14:43,385 --> 00:14:45,185
an example. In our example we are

336
00:14:45,185 --> 00:14:48,425
going to focus on AC-8 we

337
00:14:48,425 --> 00:14:49,805
picked AC-8 which is from the

338
00:14:49,805 --> 00:14:51,515

NIST special publication 800-53

339
00:14:51,515 --> 00:14:54,485
control catalog because it meets some

340
00:14:54,485 --> 00:14:56,345
key properties that make

341
00:14:56,345 --> 00:14:59,435
it very applicable for this demonstration.

342
00:14:59,435 --> 00:15:01,715
It's a simple control to understand and unlike

343
00:15:01,715 --> 00:15:03,132
many controls believe it or not it

344
00:15:03,132 --> 00:15:04,595
has a simple binary condition to be

345
00:15:04,595 --> 00:15:06,786
met the system displays the banner or

346
00:15:06,786 --> 00:15:08,315
it does not and with a simple

347
00:15:08,315 --> 00:15:10,595
binary condition like this it is most

348
00:15:10,595 --> 00:15:13,295
amenable to automated testing and the results

349
00:15:13,325 --> 00:15:18,463
of that test is easily documented and clear.

350
00:15:20,645 --> 00:15:22,145
That's not to say that there is

351
00:15:22,145 --> 00:15:24,575
only one control there are many controls

352
00:15:24,905 --> 00:15:26,555
in the 800-53 catalog

353
00:15:26,975 --> 00:15:30,011
including enhancements over 1100 of them

354
00:15:29,989 --> 00:15:34,427
The beauty of using OSCAL and actually using a profile to

355
00:15:34,595 --> 00:15:36,845
tailor and customize from the total set

356
00:15:36,875 --> 00:15:41,145
of controls in the special publication 800-53 catalog is that

357
00:15:41,135 --> 00:15:43,085
we can focus on the ones that

358
00:15:43,085 --> 00:15:45,395
we currently wish to implement against or

359
00:15:45,395 --> 00:15:47,285
the ones we want to plan or

360
00:15:47,285 --> 00:15:50,321
one or the other. The use of OSCAL

361
00:15:51,275 --> 00:15:53,465
instead of other unstructured data formats allows

362
00:15:53,465 --> 00:15:55,025
us to actually be very focused about

363
00:15:55,025 --> 00:15:56,135

what the current state of what the

364
00:15:56,135 --> 00:15:57,215
future state can be.

365
00:16:00,785 --> 00:16:03,888
In this slide here is a more thorough example.

366
00:16:03,892 --> 00:16:06,095
on the left portion of

367
00:16:06,095 --> 00:16:07,925
the slide you can see an example

368
00:16:08,345 --> 00:16:09,305
of the

369
00:16:10,715 --> 00:16:15,119
NIST SP 800-53 catalog in the OSCAL

370
00:16:15,095 --> 00:16:17,585
yaml format and you can see that

371
00:16:17,585 --> 00:16:18,965
in the old way with the traditional

372
00:16:18,965 --> 00:16:21,695
documents it's a linear document that represents

373
00:16:21,695 --> 00:16:23,345
it in a large form word file

374
00:16:23,885 --> 00:16:26,225
where the content and the representation and

375
00:16:26,225 --> 00:16:28,685
presentation of it with it's style is

376
00:16:28,685 --> 00:16:30,665
inherently linked together it makes it very

377
00:16:30,665 --> 00:16:33,067
difficult to process in part or in full.

378
00:16:35,585 --> 00:16:37,625
And so by having the

379
00:16:37,625 --> 00:16:40,053
ability to take the OSCAL catalog

380
00:16:40,086 --> 00:16:42,515
controls customize it with a profile and

381
00:16:42,515 --> 00:16:44,345
define what is currently or plan to

382
00:16:44,345 --> 00:16:46,835
be implemented in a system like the

383
00:16:46,835 --> 00:16:49,055
one in our our workflow and that

384
00:16:49,055 --> 00:16:51,065
we will demonstrate shortly you can take

385
00:16:51,065 --> 00:16:53,043
the assessment plan and read the structured

386
00:16:53,043 --> 00:16:53,585
content

387
00:16:54,215 --> 00:16:56,645
from the system security plan and the

388
00:16:56,645 --> 00:16:58,685

assessment plan and have something like GitHub

389
00:16:58,685 --> 00:17:00,516
actions using custom code that we wrote

390
00:17:00,516 --> 00:17:02,855
in this open source operate against the

391
00:17:02,855 --> 00:17:06,125
plan and execute the itemized test around

392
00:17:06,155 --> 00:17:08,375
AC-8 or other controls and then

393
00:17:09,395 --> 00:17:11,961
generate another structured result from that and

394
00:17:11,961 --> 00:17:13,355
then allow it to do the thing

395
00:17:13,355 --> 00:17:14,045
that we have

396
00:17:14,443 --> 00:17:15,935
that he can be seen in the

397
00:17:15,901 --> 00:17:18,241
interface of GitHub without looking in

398
00:17:18,275 --> 00:17:23,795
the technical logs in GitHub actions so

399
00:17:23,795 --> 00:17:26,045
we go from developer change to software

400
00:17:26,045 --> 00:17:28,895
testing to OSCAL content validation and

401
00:17:28,895 --> 00:17:31,655
then finally assessment plan execution and the

402
00:17:31,655 --> 00:17:34,505
presentation of results adjacent to a code

403
00:17:34,505 --> 00:17:35,945
change in the web interface where

404
00:17:35,945 --> 00:17:38,675
it's easily to determine the current status.

405
00:17:40,055 --> 00:17:42,095
And with that i'm going to hand

406
00:17:42,095 --> 00:17:43,775
it off to Nikita for him to

407
00:17:43,775 --> 00:17:44,308
demonstrate.

408
00:17:46,145 --> 00:17:47,525
Alright hello everyone

409
00:17:48,425 --> 00:17:50,405
if you'd like to follow along all

410
00:17:50,405 --> 00:17:53,345
code used in this demonstration as well

411
00:17:53,345 --> 00:17:56,345
as any presentation materials are available on

412
00:17:56,345 --> 00:17:59,405
GitHub under the "usnistgov" organization

413
00:17:59,855 --> 00:18:02,495

in a repository titled "blossom-case-study".

414
00:18:03,215 --> 00:18:04,655
Take a look at the pull request

415
00:18:04,682 --> 00:18:06,755
tab to see the changes will make

416
00:18:06,755 --> 00:18:10,358
during this demonstration broken down into steps.

417
00:18:12,065 --> 00:18:13,805
Here we'll be working with a simple

418
00:18:13,805 --> 00:18:14,585
application

419
00:18:15,815 --> 00:18:17,675
we have a simple web server written

420
00:18:17,675 --> 00:18:20,885
in python using FastAPI but

421
00:18:20,885 --> 00:18:22,775
the details of the application don't matter

422
00:18:22,775 --> 00:18:23,165
here

423
00:18:24,815 --> 00:18:26,675
when we run this web server locally

424
00:18:26,975 --> 00:18:28,835
we see that it produces a nice

425
00:18:28,835 --> 00:18:29,825
little web page.

426
00:18:32,885 --> 00:18:35,225
As good developers do we have test

427
00:18:35,285 --> 00:18:38,015
then verify the functional requirements of the

428
00:18:38,015 --> 00:18:40,745
system. In this case we only care

429
00:18:40,745 --> 00:18:44,015
that the root page does anything.

430
00:18:46,295 --> 00:18:49,355
We also have some CI infrastructure

431
00:18:49,355 --> 00:18:52,235
defined using GitHub actions which we

432
00:18:52,235 --> 00:18:56,045
can use to verify that developers changes

433
00:18:56,045 --> 00:18:57,575
don't break any of the tests that

434
00:18:57,575 --> 00:19:00,545
we've produced before they're merged back into

435
00:19:00,545 --> 00:19:01,145
the system

436
00:19:02,855 --> 00:19:04,505
as you can see if we switch

437
00:19:04,505 --> 00:19:06,335
to GitHub and we click on

438
00:19:06,335 --> 00:19:08,105

this most recent commit

439
00:19:09,635 --> 00:19:11,855
the tests do indeed pass.

440
00:19:16,255 --> 00:19:18,745
In our fictional organization we've been tasked

441
00:19:18,757 --> 00:19:22,237
with shifting left integrating feedback between developers

442
00:19:22,227 --> 00:19:24,927
and compliance and security early within the

443
00:19:24,930 --> 00:19:28,500
development of the fictional application. To start

444
00:19:28,767 --> 00:19:30,687
let's create a directory to house all

445
00:19:30,669 --> 00:19:31,929
of our OSCAL content

446
00:19:41,513 --> 00:19:43,943
so we've brought in a few files

447
00:19:44,016 --> 00:19:46,451
a few OSCAL files that i'll now explain.

448
00:19:48,287 --> 00:19:50,956
Firstly we have this profile.

449
00:19:50,956 --> 00:19:59,031
This OSCAL profile imports the 800-53 Rev. 5 Low Baseline

450
00:20:00,966 --> 00:20:05,526
and then it select the control AC-8.

451
00:20:08,740 --> 00:20:12,311
This produces a resolved catalog which we have here.

452
00:20:20,499 --> 00:20:23,668
Additionally we an SSP here which

453
00:20:23,655 --> 00:20:27,975
describes our system and documents how our

454
00:20:27,993 --> 00:20:30,663
system implements the controls that we've selected

455
00:20:30,829 --> 00:20:31,819
using our profile.

456
00:20:39,755 --> 00:20:42,725
Finally we have our assessment plan.

457
00:20:43,475 --> 00:20:45,695
We're using the assessment plan to shim

458
00:20:45,777 --> 00:20:48,927
information used by our custom script that

459
00:20:48,914 --> 00:20:51,494
we'll soon introduce that describes how to

460
00:20:51,516 --> 00:20:52,506
assess our system.

461
00:20:53,852 --> 00:20:55,892
The details of the way we laid

462
00:20:55,887 --> 00:20:57,777
out the assessment plan are specific to

463
00:20:57,789 --> 00:21:01,059

the script so implementers doing this your

464
00:21:01,059 --> 00:21:03,639
own way could provide the information in

465
00:21:03,629 --> 00:21:04,919
a number of different ways.

466
00:21:09,668 --> 00:21:12,728
Here as you can see we describe

467
00:21:12,804 --> 00:21:16,314
our assessment plan automation in terms of

468
00:21:16,541 --> 00:21:21,431
tasks that have associations back to a

469
00:21:21,446 --> 00:21:23,486
control through an activity

470
00:21:24,983 --> 00:21:27,563
and we also have back matter links

471
00:21:27,653 --> 00:21:30,983
to python files shall soon introduce like

472
00:21:30,989 --> 00:21:34,439
here we have this link to this

473
00:21:34,426 --> 00:21:39,064
back matter resource which is assessments/ac-8.py.

474
00:21:39,731 --> 00:21:41,771
You can see here there's additional context

475
00:21:41,800 --> 00:21:45,340
in the props we say the assessment

476
00:21:45,337 --> 00:21:48,397
result check method is the shell return

477
00:21:48,407 --> 00:21:52,217
code and the expected result is zero

478
00:21:52,544 --> 00:21:55,124
so that means we expect the script

479
00:21:55,113 --> 00:21:57,573
to run successfully in order for the

480
00:21:57,582 --> 00:21:59,382
results to be shown as

481
00:21:59,384 --> 00:22:06,258
as positive. Now let's introduce the assessments folder

482
00:22:14,166 --> 00:22:16,902
the in this assessments folder we see

483
00:22:16,902 --> 00:22:19,182
the expected ac-8.py.

484
00:22:20,605 --> 00:22:23,514
If we open it we see roughly

485
00:22:23,508 --> 00:22:26,268
what this test is doing so what

486
00:22:26,278 --> 00:22:28,348
we do in the script is we

487
00:22:28,347 --> 00:22:33,447
first try to access the website which

488
00:22:33,452 --> 00:22:34,142

is running

489
00:22:35,253 --> 00:22:38,523
in docker and once we connect to

490
00:22:38,523 --> 00:22:39,903
it we get the response and we

491
00:22:39,891 --> 00:22:42,891
parse it using an html parser and

492
00:22:42,894 --> 00:22:44,664
we tried to get the system use

493
00:22:44,663 --> 00:22:48,173
notification and assert that the system use

494
00:22:48,166 --> 00:22:51,586
notification is equal to the expected use

495
00:22:51,603 --> 00:22:56,475
notification which is defined via an environment variable

496
00:22:56,465 --> 00:22:58,167
which was passed in to

497
00:22:58,243 --> 00:23:02,473
our GitHub action. So the GitHub action

498
00:23:02,447 --> 00:23:04,967
then reads the ssp for this parameter

499
00:23:05,217 --> 00:23:07,167
passes it to the script and if

500
00:23:07,185 --> 00:23:09,015
the script runs as it's supposed to

501
00:23:09,488 --> 00:23:12,128
then the assessment results will reflect that.

502
00:23:13,392 --> 00:23:15,522
Now in order to automate this assessment

503
00:23:15,560 --> 00:23:17,180
we need to wire up a few

504
00:23:17,195 --> 00:23:19,835
custom GitHub actions that were written

505
00:23:19,831 --> 00:23:21,451
as a part of this presentation.

506
00:23:37,113 --> 00:23:41,117
the first action OSCAL validation uses

507
00:23:41,052 --> 00:23:44,189
the OSCAL CLI to validate artifacts.

508
00:23:49,011 --> 00:23:51,513
The second action takes in an assessment

509
00:23:51,496 --> 00:23:53,446
plan like the one we had just

510
00:23:53,465 --> 00:23:56,825
defined and generates an assessment results document.

511
00:24:01,006 --> 00:24:04,126
Next let's wire in our CI workflow

512
00:24:26,631 --> 00:24:30,411
To briefly go over these actions firstly

513
00:24:30,435 --> 00:24:33,315

we have the OSCAL validate action which

514
00:24:33,305 --> 00:24:36,335
consumes the reusable workflow that we adjust

515
00:24:36,341 --> 00:24:40,241
defined and it validates the profile as

516
00:24:40,245 --> 00:24:41,445
well as all the other

517
00:24:42,481 --> 00:24:44,551
OSCAL artifacts that we have defined

518
00:24:44,549 --> 00:24:45,899
in the .oscal folder

519
00:24:48,286 --> 00:24:50,776
next we have the assess action which

520
00:24:50,755 --> 00:24:54,055
only runs if the OSCAL validate action

521
00:24:54,226 --> 00:24:58,156
succeeds and if the application test as

522
00:24:58,129 --> 00:25:00,109
defined at the top the file succeeds.

523
00:25:00,499 --> 00:25:02,479
So we only assessed the system if

524
00:25:02,467 --> 00:25:03,817
we know that the system is in

525
00:25:03,802 --> 00:25:05,152
a good state to be assessed.

526
00:25:07,873 --> 00:25:09,733
This is a little more complicated than

527
00:25:09,708 --> 00:25:11,598
the rest of the actions it first

528
00:25:11,710 --> 00:25:14,470
starts the website in a docker container

529
00:25:16,314 --> 00:25:18,264
and then download all the dependencies that

530
00:25:18,283 --> 00:25:21,463
are needed and then it finally assesses

531
00:25:21,520 --> 00:25:22,180
the system

532
00:25:23,355 --> 00:25:25,815
using the assessment plan as an input

533
00:25:26,758 --> 00:25:29,938
as defined here and generating an assessment

534
00:25:29,928 --> 00:25:30,888
results document.

535
00:25:33,965 --> 00:25:37,595
Next we use the OSCAL assess

536
00:25:37,602 --> 00:25:41,772
reusable workflow yet again to validate that

537
00:25:41,773 --> 00:25:43,933
the generated assessment results is valid

538
00:25:46,645 --> 00:25:49,495

and finally we use a bit of

539
00:25:49,514 --> 00:25:54,464
custom code to comment on the commit

540
00:25:54,753 --> 00:25:57,393
with the results of the assessment

541
00:25:59,524 --> 00:26:00,364
which is done here

542
00:26:05,597 --> 00:26:08,147
so let's take this and commit

543
00:26:08,266 --> 00:26:09,436
the changes we've made

544
00:26:32,557 --> 00:26:34,567
Now if we go back to our

545
00:26:34,559 --> 00:26:35,429
pull request

546
00:26:36,461 --> 00:26:40,332
you can see that the new actions are running.

547
00:26:46,338 --> 00:26:48,318
It looks like our validation action is

548
00:26:48,340 --> 00:26:50,110
failing let's drill into why.

549
00:26:58,501 --> 00:27:00,837
Looks like we've made a mistake in our SSP

550
00:27:03,855 --> 00:27:05,593
as you can see the OSCAL

551
00:27:05,590 --> 00:27:07,991
validation action is complaining that we did

552
00:27:07,993 --> 00:27:11,534
not provide an implemented requirements key in

553
00:27:11,596 --> 00:27:14,566
the system security plan's control implementation.

554
00:27:14,555 --> 00:27:16,123
Let's add that now.

555
00:27:31,249 --> 00:27:34,039
"implemented-requirements" allow us to specify how

556
00:27:34,052 --> 00:27:37,112
we implement control here you can see

557
00:27:37,122 --> 00:27:40,212
AC-8's param one defines the system

558
00:27:40,191 --> 00:27:43,431
use notification banners expected text.

559
00:27:45,930 --> 00:27:48,570
Automated assessment scripts such as the one

560
00:27:48,566 --> 00:27:51,206
we have defined for AC-8 can validate

561
00:27:51,202 --> 00:27:53,438
our system against these parameters.

562
00:27:53,443 --> 00:27:56,012
Let's commit the fixed ssp now

563
00:28:29,240 --> 00:28:32,750

Looks like the application test and OSCAL

564
00:28:32,777 --> 00:28:36,715
validate actions completed successfully, however

565
00:28:36,698 --> 00:28:40,602
the OSCAL assess action did not. And it also

566
00:28:40,585 --> 00:28:43,688
looks like a comment was left on the commit.

567
00:28:44,622 --> 00:28:46,068
If we click on the comment we can

568
00:28:46,091 --> 00:28:49,601
see that the automated assessment results failed

569
00:28:57,967 --> 00:29:00,436
the clicking on more details of the action,

570
00:29:01,773 --> 00:29:03,843
and going to the summary we can

571
00:29:03,842 --> 00:29:06,962
also see the produced assessment results file

572
00:29:09,247 --> 00:29:11,047
which is a valid OSCAL document.

573
00:29:13,475 --> 00:29:15,610
The reason that our system is not

574
00:29:15,587 --> 00:29:19,907
compliant is that the notification banner currently

575
00:29:19,924 --> 00:29:23,044
says "enjoy your stay" which is not

576
00:29:23,027 --> 00:29:24,227
the expected result.

577
00:29:26,331 --> 00:29:28,101
This can easily be fixed however.

578
00:29:31,536 --> 00:29:33,246
If we go back into the application

579
00:29:33,772 --> 00:29:36,862
looks like i accidentally put the nonconforming

580
00:29:36,875 --> 00:29:39,215
banner which looks

581
00:29:40,445 --> 00:29:41,195
like this.

582
00:29:43,166 --> 00:29:46,803
Now this can easily be fixed by swapping

583
00:29:46,818 --> 00:29:48,348
in the conforming banner.

584
00:29:49,721 --> 00:29:52,121
So we go back into our application

585
00:29:53,424 --> 00:29:55,074
when you type in conforming

586
00:30:08,640 --> 00:30:10,260
now if we reload the page

587
00:30:11,976 --> 00:30:12,996
that looks more like it

588
00:30:16,548 --> 00:30:19,008

we can now commit the system which

589
00:30:18,983 --> 00:30:20,423
has been brought into compliance

590
00:30:51,082 --> 00:30:52,762
In a few short minutes we've taken

591
00:30:52,784 --> 00:30:55,814
an existing application and added automated assessment

592
00:30:55,787 --> 00:30:58,847
capabilities to it. The workflow used to

593
00:30:58,857 --> 00:31:01,107
accomplish this is available for anyone to

594
00:31:01,125 --> 00:31:02,295
reuse and improve.

595
00:31:04,429 --> 00:31:06,799
Thank you for watching this presentation we

596
00:31:06,798 --> 00:31:09,918
welcome any feedback enhancements and questions in

597
00:31:09,901 --> 00:31:11,731
the project GitHub repository.