

1

00:00:11,346 --> 00:00:18,786

Here are four demos that i hope to share with you. The first one as Anca

2

00:00:18,786 --> 00:00:27,346

has explained is about the OSCAL exchange protocol that we've developed and are using for our SSP

3

00:00:27,346 --> 00:00:34,066

um and this is an idealized version um that we've come to support

4

00:00:34,706 --> 00:00:41,346

based on upon our real life experience with the SSP and I have two links here one is um

5

00:00:42,066 --> 00:00:49,186

uh the code that you can download and run yourself and try it and the second is is a live demo that

6

00:00:49,186 --> 00:00:54,466

is also uh publicly available and you can go there and give it a try this is actually kind of

7

00:00:54,466 --> 00:01:03,786

a work in progress it's not complete and not fully functional but it sort of works so let's see

8

00:01:12,306 --> 00:01:20,946

so here is the GitHub repo that has the OSCAL exchange protocol code and you can go here

9

00:01:20,946 --> 00:01:26,306

and it explains what you minimally have to do to actually run it you just need to have a reasonable

10

00:01:26,866 --> 00:01:37,026

uh python and then you can just clone it and run it and so here um

11

00:01:40,626 --> 00:01:50,626

here is the example that's running live that you can go to as well and and look at um and um as uh

12

00:01:51,426 --> 00:01:58,066

i'll first say that this is um based on openAPI and i'm using fastAPI as the as the

13

00:01:58,066 --> 00:02:06,226

tool to implement this and uh as Anca mentioned, we have these uh different um life cycle

14

00:02:06,226 --> 00:02:12,466

operations to to communicate with the scc and um so we don't envision that each individual

15

00:02:14,946 --> 00:02:20,146

OSCAL document is going to be edited using this tool we envision that we're just going to deploy

16

00:02:20,146 --> 00:02:26,226

and manage whole documents to the scc who's going to then use those to to do their business

17

00:02:27,106 --> 00:02:32,466

and so what we have is actually a life cycle operation for each of the OSCAL documents that

18

00:02:32,466 --> 00:02:39,906

line up one for one for the different OSCAL documents and then down here we have um two

19

00:02:39,906 --> 00:02:47,346

kinds of validation things one validation thing is for uh policy validation points to actually

20

00:02:47,346 --> 00:02:53,586

get their checks for example and configure themselves to run the checks and then post

21

00:02:53,586 --> 00:02:59,266

results and so we have this validation phase and we have this validation in two phases currently

22

00:02:59,986 --> 00:03:06,626

we're using profiles and so we're kind of in phase one for rscd in the future um we hope

23

00:03:06,626 --> 00:03:13,506

to use system security plans as the as the basis documents and so uh what uh what a pvp would do is

24

00:03:13,506 --> 00:03:18,866

it would come here and say i'm this pvp please give me the set of profiles that i should be

25

00:03:18,866 --> 00:03:28,946

concerned about and uh there's also some filtering allowed where you can specify different uh

26

00:03:28,946 --> 00:03:33,906

components or other kinds of things that you want to filter so you don't get everything uh

27

00:03:34,706 --> 00:03:39,266

if that's what you want to do and then when you have your results after you're running your checks

28

00:03:39,266 --> 00:03:44,466

you want to give them back into the the security compliance center and so we have a post that says

29

00:03:44,466 --> 00:03:48,866

here are my results and here's the system security plan that these results are associated with

30

00:03:50,306 --> 00:03:53,586

um so that's a whirlwind tour of what the uh

31

00:03:55,026 --> 00:03:59,106
exchange protocol is i don't know if Anca wants
to mention anything else or if i should move on

32
00:03:59,106 --> 00:04:05,346
to that just to address a very important question
uh on the eu ids how do we manage the year ideas

33
00:04:05,346 --> 00:04:12,466
so obviously if we have a full uh framework
implementation right we are able to exchange uh in

34
00:04:12,466 --> 00:04:18,146
in in those um in the in the protocol the
uuids together with the body of the of the

35
00:04:20,786 --> 00:04:29,426
information that is sent since we started um in
into a phasing with a phasing approach to adopt

36
00:04:29,426 --> 00:04:37,106
OSCAL and the uh uh the framework uh initially
we do not have uuids so when we send a profile

37
00:04:37,106 --> 00:04:43,746
or we send the SSP structure uh the the uuid
is of course is a you know it is fake just to

38
00:04:43,746 --> 00:04:52,386
have a valid uh fully valid OSCAL formatted
artifact but as we mature and we start now

39
00:04:53,506 --> 00:04:56,866
having the end-to-end support for all the elements

40
00:04:56,866 --> 00:05:05,266
right obviously the request for results will
contain an SSP with a valid uh SSP uuid and when

41
00:05:05,266 --> 00:05:12,146

the results passes back the assessment results
that will contain it so this is this is a generic

42

00:05:13,506 --> 00:05:19,986
recommendation that we have for uh who wants
to start her doctor OSCAL right start in

43

00:05:19,986 --> 00:05:27,106
phases and as you imagine and consider all the
elements but you know you reach the maturity uh

44

00:05:27,106 --> 00:05:35,026
to really using them only when you have the
full um the full framework uh implemented

45

00:05:35,666 --> 00:05:35,906
okay

46

00:05:38,786 --> 00:05:41,906
um so the next thing i was
going to show very quickly is uh

47

00:05:41,906 --> 00:05:50,626
actual installation of trestle so um you could go
to this link and it explains how um to do that and

48

00:05:51,426 --> 00:05:56,226
so this is this is it it's pretty simple
to do and i'll try and do it quickly

49

00:06:00,626 --> 00:06:03,026
oops

50

00:06:24,226 --> 00:06:31,026
sorry about that

51

00:06:50,066 --> 00:06:50,226
all

52

00:06:50,226 --> 00:07:02,706
right so i'll just show you this um so we just um

53
00:07:03,826 --> 00:07:09,506
create a virtual environment and source it and
then we go ahead and say python install and

54
00:07:11,586 --> 00:07:13,826
upgrade pip and then

55
00:07:16,226 --> 00:07:17,186
we install trestle

56
00:07:21,506 --> 00:07:21,906
and then

57
00:07:24,146 --> 00:07:31,426
you're able to say uh trestle um list the
commands that you're able to execute um

58
00:07:31,986 --> 00:07:36,786
so um i don't know what i did much to myself in
setting up for this but i screwed myself up and i

59
00:07:36,786 --> 00:07:43,346
apologize for that um and uh the next
thing i want to show really quickly

60
00:07:43,346 --> 00:07:50,306
is uh the trestle markdown for uh SSP um and
this is um with Vikas and frank suits and

61
00:07:50,306 --> 00:07:59,186
uh Ekaterina Nikonova from IBM Australia and
here's a link that where this demo is located

62
00:07:59,186 --> 00:08:04,786
and you can navigate to this from the trestle
uh website um so let me see if i can go there

63

00:08:06,066 --> 00:08:12,146

so this is the demo and as Vikas mentioned earlier you know we envisioned that different people with

64

00:08:12,146 --> 00:08:23,106

different skills um are going to be uh wanting to contribute to the OSCAL process and so in um

65

00:08:25,106 --> 00:08:34,226

uh in this in this uh imagined scenario um we have um in pink um OSCAL uh objects and in blue

66

00:08:34,226 --> 00:08:42,066

um markdown representations and so we as uh as um Vikas described we can go back and forth between

67

00:08:42,066 --> 00:08:48,066

these things um and then at the end of the day um we're gonna have uh go ahead and produce a

68

00:08:48,066 --> 00:08:56,546

a markdown of an SSP and then finally a a markdown word document that would be suitable for a um

69

00:08:58,626 --> 00:09:03,106

auditor to view um so um seeing that i only have two minutes left

70

00:09:03,106 --> 00:09:07,266

i'm actually not going to run that demo but i'm just going to go ahead and um

71

00:09:09,426 --> 00:09:13,266

quickly show the last thing i wanted to show and as again as we mentioned

72

00:09:13,266 --> 00:09:18,226

um earlier uh some people are more comfortable with spreadsheets than then

73

00:09:20,946 --> 00:09:28,306

using json directly and so again this is navigable from the from the um trestle based website

74

00:09:28,946 --> 00:09:37,666

this is a demo that shows uh how a spreadsheet can be converted into a pascal document particular

75

00:09:37,666 --> 00:09:47,346

component definition and so um this is what a what the spreadsheet would look like to the to the CIS0

76

00:09:47,346 --> 00:09:54,946

for example uh where they could have different rules and missed mappings and resources and

77

00:09:57,186 --> 00:10:02,866

what comes out the other end would be a component definition um

78

00:10:02,866 --> 00:10:10,466

that um represents what uh that spreadsheet uh contained and again because i have no time left

79

00:10:10,466 --> 00:10:13,586

i'm not actually gonna show the demo i'm gonna give it back to Anca if she wants to wrap up

80

00:10:14,146 --> 00:10:18,226

thank you oh thank you Lou this is this is very good i think uh

81

00:10:18,866 --> 00:10:28,066

there are a lot of valuable um tutorials and uh demos that you can follow on the on the on the

82

00:10:28,066 --> 00:10:35,906

website so thank you all for sharing the links and with that i'll give it back to Michaela

