```
1
00:00:00,000 --> 00:00:08,000
, thank you. OK, great. So again, Robert McHale and the company's
policy workgroup

2
00:00:08,000 --> 00:00:18,000
co-chair and also a member of Very Sincere and communities, segues
focusing on security. And I do have a day job as CTO of Sunstone
Secure where we do work

3
00:00:18,000 --> 00:00:28,000
with just four years and three powers and agencies on etc. So that's
where I bring that personal experience, as Michaela mentioned. So
today, as we've been participating

4
00:00:29,000 --> 00:00:38,000
in the Moscow community, we have seen a lot of presentations on the
CSP perspective or the catalog perspective and understanding the
controls and how to

5
00:00:38,000 --> 00:00:49,000
mouth those two components. We hadn't seen a lot of presentation on
some from an agency perspective. How would we actually use Moscow

6
00:00:48,000 --> 00:00:58,000
beyond just getting documentation in a new format? So that's who I'm
addressing today. But everybody is, of course, welcome. And in

7
00:00:58,000 --> 00:01:08,000
prepping for this, I promised I'd keep it kind of level one to level
three. So if you've never heard of Moscow, we'll start with a little
bit of an intro and then we'll kind of ramp

8
00:01:08,000 --> 00:01:18,000
up the technical detail and I promise I can keep on time. We're going
to have a little demo and then everything I am showing is, of course,

9
00:01:18,000 --> 00:01:28,000
our opinions. It is not represented by ness that is not endorsed by
Federation for any U.S. government agency. So please

10
```

```
00:01:28,000 --> 00:01:38,000
, we will post all of these materials and any examples we show on an
open GitHub repo. So everybody is welcome to download them, use them

11
00:01:38,000 --> 00:01:47,000
, represent them any part of this? I will say that our experience with
agencies and working through the federal process

12
00:01:47,000 --> 00:01:57,000
is that it is very, today very document focused. So your support and
some of these terms and concepts come directly from the veterans
training

13
00:01:57,000 --> 00:02:07,000
. You're looking for inconsistencies and in the control narratives,
the diagrams and how those match up to the the narratives. You know,
the system security plan

14
00:02:08,000 --> 00:02:17,000
and the three power test cases, you know, there's a lot of work that
makes sure everything's in alignment. I will highlight that there are
these relationships embedded in the work

15
00:02:17,000 --> 00:02:27,000
activities for an agency and how they're thinking through the process
of digesting all of this documentation. And so I'll highlight those
relationships

16
00:02:26,000 --> 00:02:29,000
in a little bit

17
00:02:29,000 --> 00:02:39,000
. I would say it's a lot of work for an agency, I think, you know, or
a lot of us work with spouse, or maybe they are spouse, work with a
lot of three powers

18
00:02:39,000 --> 00:02:49,000
. And I think what gets lost in the conversation and there's a lot of
work on the agency side to either sponsor a bedroom or even reuse an
existing

19
```

00:02:49,000 --> 00:02:58,000
for and package, right? They have to understand the impact their systems and the interconnections they have to understand, you know, the risks

20
00:02:58,000 --> 00:03:08,000
associated with the system and the controls and how they're implemented. You know, they have to try to digest a totally new system policy and procedures and make sure that those are covering all

21
00:03:08,000 --> 00:03:18,000
the statements after says, you know, whether the three powers doing their job while they're accredited, you know they have to they have to take ownership that the tests were correct

22
00:03:18,000 --> 00:03:28,000
. And of course, they're signing up for a long term responsibility if they're the sponsoring agency, especially that they're going to monitor and review online every month. Continuous monitoring

23
00:03:28,000 --> 00:03:38,000
. So it's a lot of work. And I think, you know, as a community, we owe it to the agencies to try to present a path forward that, you know, reduces their effort and

24
00:03:38,000 --> 00:03:48,000
in the end, if we can make this re-use possible with a lot less effort on the agency part, we can really accelerate the timeline

25
00:03:48,000 --> 00:03:58,000
to getting agencies to sponsor new federal packages or reuse existing federal packages, and that supports innovation. And

26
00:03:57,000 --> 00:04:08,000
so those who may not be familiar with federal and the agency work, you know, just to cover briefly, they have to understand all the data and the data flows and the sensitive

27
00:04:08,000 --> 00:04:18,000
issue of that data. They have to kind of map that up with the documentation that digital identity and the strength of that identity that we have to work through. And, you know, very

28
00:04:18,000 --> 00:04:27,000
cumbersome word document and excel spreadsheet process to do the
analysis and review. And then they have to digest the three panels

29
00:04:27,000 --> 00:04:37,000
, assessment results and review. So and then of course, there are
multiple layers to any complex system these days. So you're usually
leveraging somebody else's into

30
00:04:37,000 --> 00:04:47,000
a cloud provider, another app or connecting to other federal company
authorized systems. So a lot of work and then it takes a team. So

31
00:04:47,000 --> 00:04:56,000
, you know, there's agencies we work with. There's a number of people
on each of these calls. There's a lot of work happening in the
background. So there's probably, you know, half

32
00:04:56,000 --> 00:05:06,000
a dozen to a dozen folks actively working. If you're if they're going
through the sponsorship process and then even if they're using NATO,
there's still a lot of that review and

33
00:05:06,000 --> 00:05:16,000
GRC work that has to be done. And there's not a lot of like this
letter. There's not a lot of clear definition of coverage like

34
00:05:15,000 --> 00:05:25,000
documentation that we're reviewing and all of the controls really
covering all of the system threats. And we'll talk about threats and
coverage later

35
00:05:25,000 --> 00:05:35,000
on my presentation. Um, so again, just to summarize, you know, a lot
of manual operations kind of ad hoc mapping between, you know, the
CCP's interpretation of the

36
00:05:34,000 --> 00:05:44,000
controls and the underlying system, you know, the maybe, you know,
deviations that need to be discussed there is going to inheritance

from different Asian systems

37
00:05:44,000 --> 00:05:54,000
and you know how how those changing traceable over time. Again, the three powers using some sampling methodology, but even for a specific control or even a control part,

38
00:05:54,000 --> 00:06:04,000
they're recovering the intent of that control relative to the system. You need a lot of subject matter experts to assess these risks, and then you've got to worry about ongoing

39
00:06:05,000 --> 00:06:15,000
things like incident response preparations, the recovery programs and then every year happens again, annual requirements for better and then significant changes

40
00:06:15,000 --> 00:06:24,000
again. Just to summarize some of the feedback we've heard directly from agencies on why they either hesitate to sponsor an NCO

41
00:06:25,000 --> 00:06:34,000
or around the package or even to reduce the federal package adds burden. And typically, it's around the staffing and the time it takes

42
00:06:34,000 --> 00:06:44,000
. But as I note, you know, adversaries love status quo. So if we're not innovating, we're lagging behind the adversary. So

43
00:06:44,000 --> 00:06:54,000
was OK. Oh, and one of my favorite pet peeves is that there's those of you who work through kind of the Conran process, you know, everything kind of to R5 for vulnerabilities

44
00:06:54,000 --> 00:07:04,000
are detected. So we'll talk about this later. But I think we want I want to think from the agency's perspective of

45
00:07:04,000 --> 00:07:13,000
what we're really trying to do is understand the threats to the system and, you know, vulnerabilities and controls and how those are

interrelated. So

46
00:07:13,000 --> 00:07:23,000
we are not alone in presenting a vision for accelerating and reducing the work. And this was from a presentation in February 2021. There's an explicit

47
00:07:23,000 --> 00:07:33,000
goal to really condense the timeline, reduce the work and really focus on risk management and less on checking boxes. So I think the PMO

48
00:07:33,000 --> 00:07:43,000
and certainly the folks at Nest have declared clearly that this is a goal. So together, let's talk about how IOSCO helps roll up our sleeves and get

49
00:07:43,000 --> 00:07:52,000
the work done faster and with more threat and risk in mind. So clarity. So you know, we're writing things in code with IOSCO defined set of schemas

50
00:07:52,000 --> 00:08:02,000
based on very clear models. So we're not, you know, looking at things, we're trying to make things clear. We're literally writing code that reads

51
00:08:02,000 --> 00:08:12,000
data and producing results. We go from kind of manual checklists and excel templates to peers. If

52
00:08:12,000 --> 00:08:22,000
you're familiar with. We'll talk a little bit more about kit and get ops. Everything is a automatable process in a repository that has automated workflows

53
00:08:21,000 --> 00:08:31,000
, right? It is a team sport and everybody needs a different view of the problem. Some only need all the details and all the interconnections

54
00:08:32,000 --> 00:08:41,000

, so we'll just need a summary of you and want that word document or the one a PowerPoint presentation. ASCO allows code to slice off the bench that it

55
00:08:41,000 --> 00:08:51,000
needs and present only the information at the right level of detail for that audience. And consistency is probably the most those of us who

56
00:08:51,000 --> 00:09:01,000
actually work with IOSCO and the schema. Everything is identified. Everything is linked together. Everything is traceable across the different models

57
00:09:01,000 --> 00:09:11,000
. So we try to remove all of the inconsistencies and all the ambiguity and making sure that everything is always lined up as things change

58
00:09:10,000 --> 00:09:20,000
. I think one thing that is underappreciated, perhaps, is that the current process is very linear, and IOSCO by itself doesn't necessarily

59
00:09:21,000 --> 00:09:30,000
change that. But it enables a more parallel process where you can now have multiple teams working on a shared set of artifacts

60
00:09:30,000 --> 00:09:40,000
in a great repository that is version controlled, arm back controls and auditable. And now you can have parallel branches and parallel workflows and

61
00:09:39,000 --> 00:09:49,000
automated, you know, GitHub actions if you're familiar with. And they're very usable, so you can very quickly clone across groups and as individuals

62
00:09:50,000 --> 00:09:59,000
come on and come off the project. And so you get a lot of parallel processing that all converges where it needs to. And I'll have another graphic that tries to convey a bit more of that

63
00:09:59,000 --> 00:10:09,000
later. Budget always important. So, you know, directly related to staff time. So everybody's, you know, twenty five folks on a on a call to review

64
00:10:09,000 --> 00:10:19,000
. And you know, they may be subject matter experts in a particular slice of it. That's a lot of resources for each of those meetings. Whereas if we can make this more asynchronous

65
00:10:18,000 --> 00:10:28,000
and have subject matter experts focus on distinct parts and capture that in code or in rules which we'll talk about shortly. Then you really can

66
00:10:29,000 --> 00:10:38,000
reduce that timeline in the budget. And I think a challenge as part of an agency discussion last year about the biggest challenge is staffing

67
00:10:38,000 --> 00:10:48,000
. So those are leaving. Folks are not trained. So this is really what keeps a lot of agency and mission leaders up at night. So

68
00:10:48,000 --> 00:10:58,000
if you can start to embrace O'Scanlon automation, you're now force reinforcing new skills and a new culture that really excites

69
00:10:58,000 --> 00:11:08,000
. I hesitate to say the millennials, but the younger workforce and you know, that's what they're exposed to in their education, and that's what they're looking

70
00:11:08,000 --> 00:11:18,000
for in their career path. And so you're building this skills that will help you recruit that talent, retain that tone and then the asterisk courses you really do need leadership

71
00:11:17,000 --> 00:11:27,000
, sponsorship and engagement for this to work for for obvious reasons. So our skill helps again with the consistency

72
00:11:28,000 --> 00:11:37,000
and then the verification. So we have the ability to write validations and we won't go into those because a lot of folks who covered

73
00:11:37,000 --> 00:11:47,000
the validation rules and how to make sure that all of the parts of school are in the dark and the data presented is open source tools and a lot of commercial

74
00:11:47,000 --> 00:11:57,000
vendors working on solutions. So we'll cover a bit beyond that. But we want to kind of understand how we can map a security capability

75
00:11:57,000 --> 00:12:06,000
, encode in Moscow to a control implementation. So I'm sure that in the demo we want, we won't show the profiles and variables, but it's important

76
00:12:06,000 --> 00:12:16,000
concept. Others have talked about it that these are documents that can be tailored to an agency's risk model and risk appetite and

77
00:12:16,000 --> 00:12:26,000
requirements. And then we want to have that full and mapping from a component control capability threat. And then this really enables reinforces

78
00:12:26,000 --> 00:12:36,000
that risk management mentality. And as I'll show you, it also helps you define coverage metrics at all, at all levels and

79
00:12:36,000 --> 00:12:46,000
this is going to require a bit of kind of a dual perspective. So on one side of our scale is what I'll call abstract classes. So you've got catalogues

80
00:12:47,000 --> 00:12:56,000
, you've got component definitions, right? And those of us who might come from above to join in programming will understand this notion of an abstract class, maybe an inheritance that you

81
00:12:56,000 --> 00:13:06,000
interface if you come from a Java background. And then on the other side, you've got objects, object instances, implements and methods. So

82
00:13:06,000 --> 00:13:16,000
you've got this notion of a real thing that can be compiled or deployed, right? And then in a writing system, you've got an inventory and you've got the actual stuff that you're

83
00:13:16,000 --> 00:13:26,000
trying to use and assess. So you want to understand what is my risk, not only at the abstract level. So have

84
00:13:25,000 --> 00:13:35,000
I define all of the controls that I need to define for a particular baseline or profile? And how do those all map together, but also about the implementation

85
00:13:36,000 --> 00:13:46,000
level? So if I build the system according to that schema to that abstract class and implement those things that I need to implement, have I covered

86
00:13:45,000 --> 00:13:55,000
all the risks? And then the running system, of course, which I think a lot of us with its security background we focus on with the day to day where, you know, we're scanning things where

87
00:13:55,000 --> 00:14:06,000
were enforcing controls at a technical level and configuration level. So that's kind of the thing that you work with day to day. That's the runtime

88
00:14:05,000 --> 00:14:10,000
and coverage at that level is a little bit easier

89
00:14:10,000 --> 00:14:20,000
. So let's say how you go. So risk assessment. So the benefits of ASCO. Right. So the validations, those kind of schema

90

00:14:20,000 --> 00:14:29,000
channels a check, hey, is my ask out complete and have all the fields in here? And is it all the links? That's going to eliminate your SSP inconsistencies

91
00:14:29,000 --> 00:14:39,000
. So that car's off a huge amount of work for the agency component and controls code. You know, hopefully my strong opinion is

92
00:14:39,000 --> 00:14:49,000
that we can deprecate the diagrams and the narrow text. They can be useful as humans consume the results of the assessments and

93
00:14:48,000 --> 00:14:59,000
they can certainly provide contextual information for the, you know, giving you situational awareness. But hopefully we can in the not too distant future, stop relying

94
00:14:59,000 --> 00:15:09,000
on them as the definition of completeness or coverage rules, which I'll I'll talk to. Not yet an official ASCO

95
00:15:09,000 --> 00:15:19,000
schema. I know there's a lot of work going on. I think I put the GitHub issue later on really more about the tests of the control. The

96
00:15:19,000 --> 00:15:28,000
control implementation, less about the consistency and the validation. So actually, how are we going to test this in code to make sure that the control and

97
00:15:28,000 --> 00:15:38,000
the components implementing these requirements actually match the intent? How can it demonstrate that and all of this to bring it back to that risk management is about, you know, she

98
00:15:38,000 --> 00:15:48,000
deposits if you're familiar with the major attack model, but there are other models that kind of rely on threat for procedures and an indicator

99

00:15:48,000 --> 00:15:58,000
of threats. So all of that helps us model these risks and the
relationships between them. So how do we do this? So how are we going

100
00:15:58,000 --> 00:16:07,000
to actually implement this as an agency, first and foremost is again,
that culture and mentality shift? You know, we should expect Moscow.
We should

101
00:16:08,000 --> 00:16:18,000
embrace Moscow, we should use it. We shouldn't see it as kind of an
appendix that, you know, maybe one day we'll use. We really need to
find the champion that can say, this is where we're

102
00:16:17,000 --> 00:16:27,000
going. We understood the risks, the benefits, the costs and let's do
that. And to support that, certainly since you have some of the
sincere

103
00:16:27,000 --> 00:16:37,000
participants, our Communities Policy Work Group all providing open
source tools, help and support community support and a lot of the
folks on this call providing

104
00:16:37,000 --> 00:16:47,000
open source tools, community support. And obviously there are vendor
commercial solutions to but things in Seattle and communities

105
00:16:47,000 --> 00:16:53,000
oriented. You know, everything I'm talking about is open source today.

106
00:16:53,000 --> 00:17:03,000
that reskilling so changing who is involved in the skill sets, so you
may still have the authorizing official. Hopefully that person is the
champion

107
00:17:03,000 --> 00:17:13,000
and the sponsor for this change. But instead of the subject matter
experts being, you know, narrative oriented, now you're looking at you

108
00:17:13,000 --> 00:17:23,000

know, code pipelines and code artifacts, right? And so you're going to have all the ASCO for your SSP, your SOP, your online validations, your controls

109
00:17:23,000 --> 00:17:33,000
and in some code repo and just call it get on the other side, you're going to have your subject matter experts writing rules, and I'll show you some examples of that. And then there's policy

110
00:17:33,000 --> 00:17:42,000
rules as configuration checking rules. There's threat mapping rules becoming artifacts that allow this automated Oscar pipeline to work

111
00:17:42,000 --> 00:17:52,000
. So again, hammering on the documentation issue, using algorithms and so, you know, drawing from commercial world experience

112
00:17:52,000 --> 00:18:01,000
, this is already happening. So a lot of commercial implementations of compliance and security are no longer relying on documentation checklists or

113
00:18:02,000 --> 00:18:11,000
documented narratives on how compliance works. They're using algorithms to define, you know, how how things are interconnected, and I'll show a

114
00:18:11,000 --> 00:18:21,000
picture of that shortly. They're looking at the strength of those connections that are tagging things with data and and then using algorithms to measure

115
00:18:21,000 --> 00:18:31,000
. And you calculate, are we covering all the controls? What is the strength and intensity? What are the rules associated that need to pass and how specifically

116
00:18:30,000 --> 00:18:40,000
can you test that? And then again, embrace this notion of automating everything. Don't see automation as a nice to have. If this is going to work

117
00:18:41,000 --> 00:18:50,000
, you really need to automate as much as possible. So think of it
again as compliance is code, you know, get ops and the ability to make

118
00:18:50,000 --> 00:19:00,000
a change in a repo that's very controlled with usable, auditable. And
when you commit, that goes through a pipeline code process and makes
changes available

119
00:19:00,000 --> 00:19:10,000
to others that need to participate or use those artifacts even down to
the final stop. This is a future view where you can imagine

120
00:19:10,000 --> 00:19:20,000
having a final authority to operate PR that the authorizing official
approves, and that can trigger downstream

121
00:19:20,000 --> 00:19:30,000
workflows that the system itself can use as a signal for turning on
different features. If you're familiar with the notion of like feature
flags. You can imagine having an NCO trigger

122
00:19:30,000 --> 00:19:38,000
features like this is now. These systems and services are available
with with the requirements

123
00:19:38,000 --> 00:19:48,000
all satisfied. And then again, I mentioned earlier this notion of
parallel workflows those who work in software. I know the git flow
process is going to be very familiar. So

124
00:19:48,000 --> 00:19:58,000
you instead of everybody kind of being involved at every point, you
know, looking at the components, evaluating the security capabilities,
how do we tested

125
00:19:58,000 --> 00:20:08,000
Jimmy pull on the right controls? Are the controls mapped correctly?
So having everybody on those calls and on those email threads really
can break this out into I'm the subject

126
00:20:08,000 --> 00:20:17,000
, not in my part. I'm going to participate where I need to modify that
and get merge it into the feature. Branches view well and then let my

127
00:20:17,000 --> 00:20:27,000
my team take it from there. And of course, I can monitor it. I get I
get notifications in the repo. I can always review the chatter in the
comments. Just as all of you are probably doing

128
00:20:27,000 --> 00:20:37,000
in the the Nest GitHub for asking itself. So it all becomes very
codified. It all becomes pipeline and it all at the very end

129
00:20:37,000 --> 00:20:47,000
that to bring in just the right people at just the right time. So a
big view of a component of how I view this

130
00:20:47,000 --> 00:20:57,000
problem is derived from this diet 811, which is a great publication.
If you haven't read it, you should, but it really talks about a

131
00:20:56,000 --> 00:21:06,000
desired state and actual state view of the world. So we're going to
use, you know, in this day are they're not even talking about our
scale, they're just saying we're going to document

132
00:21:07,000 --> 00:21:17,000
what our desired state is, what security capabilities we need to meet
these controls. And then we're going to look at the actual state and
do tests to make sure that the actual

133
00:21:16,000 --> 00:21:26,000
state matches. I even started one new concept is what I call declared
state. As we move into a more code

134
00:21:27,000 --> 00:21:36,000
systems thinking view of the world, we're going to split out. Here's
my desired state I may have defined for a control catalog

135
00:21:36,000 --> 00:21:46,000

, a profile, a baseline. I'm now going to translate or complement that with declared rules and those rules from the subject matter experts are going

136
00:21:46,000 --> 00:21:56,000
to support the implement in some cases, even determine the implementation of those controls. And then, of course, all of that connects to the actual state

137
00:21:56,000 --> 00:22:06,000
. And so the crux for looking at the world in this way. You have to have policy rules. So I'll show you a demo of rigor

138
00:22:06,000 --> 00:22:16,000
, which is open source implementation of a rules engine. Regan's language open policy is the enforcement open policy agent

139
00:22:16,000 --> 00:22:26,000
is the enforcement engine. You need to have a model for risks and threats. So threats, minor attack minded defense

140
00:22:25,000 --> 00:22:35,000
is the instance that I will rely on. There are others, but you really do have to have a working definition of the threats to your system and then you have to have

141
00:22:35,000 --> 00:22:45,000
test. And what I call query rules so that you can map your intent to test that can run against the

142
00:22:45,000 --> 00:22:55,000
actual inventory of things. And as I mentioned, my my view of the future is that, you know, ATO is code that actually can be part and and trigger things downstream.

143
00:22:54,000 --> 00:22:57,000
The running system

144
00:22:58,000 --> 00:23:07,000
so what is policy is code. So for those of you who haven't seen this and Kubernetes or other systems, it's essentially declarative rules

```
145
00:23:07,000 --> 00:23:17,000
that state the intent of a particular. It could be, you know,
authorizations are about what permissions I have, and it has a very

146
00:23:17,000 --> 00:23:26,000
lightweight enforcement engine that can evaluate all of the different
rules logically, potentially Boolean operations and make a
determination

147
00:23:26,000 --> 00:23:36,000
. However, one thing that folks don't know about this policy is code.
Well, I'll I'll get to that next. One thing that isn't obvious

148
00:23:36,000 --> 00:23:46,000
from this policy is code engines is they can actually generate code.
So I'll show you a demonstration of that later. So there has been
threat guidance

149
00:23:46,000 --> 00:23:55,000
, just an aside quickly from the PMO on how to look at controls for an
agency lens, mapping that to a risk model.

150
00:23:55,000 --> 00:24:05,000
So you want to look at what is the protection? They have a scoring
rubric to how you know the control protection

151
00:24:05,000 --> 00:24:15,000
against these risks and of course, a coverage metric about how, how
well it's implemented and that allows you to prioritize the controls
you're going to assess.

152
00:24:15,000 --> 00:24:24,000
So that it matches your your risk profile. So now we have our desired
state. We have a plan that talks about

153
00:24:25,000 --> 00:24:35,000
in Moscow our actual task for that zone state and we can map threats
through security. KTLA Wildflower

154
```

00:24:34,000 --> 00:24:44,000
and for those mitigations. Can you please mute yourself? I'm sorry. Wildflower CEO is looking to set up a call for Friday

155
00:24:44,000 --> 00:24:51,000
, and then you might want to have folks mute. It's just like

156
00:24:52,000 --> 00:25:01,000
, OK, thank you. So I do highlight open policy agent and that's what we use. But there are other agents. Sorry,

157
00:25:01,000 --> 00:25:11,000
policy enforcement engines. Another way to implement this commercially we use this methodology is we do use a graph database and run graph queries

158
00:25:11,000 --> 00:25:20,000
. And so that can actually make decisions based on the relationships and the properties of each entity in a in a graph. And then if you've if

159
00:25:21,000 --> 00:25:30,000
you're really bored, you know, we've done some modeling around this and the policy work group, but you can use things like formal verification and S.A.

160
00:25:30,000 --> 00:25:40,000
and solvers. And so you can really get to formal logic definitions of the rules and the underlying system. But

161
00:25:39,000 --> 00:25:45,000
for today, we'll use the lightweight approach with open policy. Um,

162
00:25:45,000 --> 00:25:55,000
so for under understanding how the threats are being addressed, you do need to have a query on the actual system, right? So there

163
00:25:55,000 --> 00:26:04,000
are a few different approaches to this. We've used most of them and there are others that are out there in the research and open source, so you can use the

164
00:26:04,000 --> 00:26:14,000
natural language in the narratives and the controlled definitions and the supplemental guidance and the policies and procedures in the component documentation

165
00:26:14,000 --> 00:26:24,000
. And then you can use semantic matching or NLP entity extraction. And you know, if you will, keyword matching to try to to automate that mapping so that you can say

166
00:26:24,000 --> 00:26:34,000
, sure, you know, the threat language and my implementation language, do they match up so you can build statistical models and you can measure in

167
00:26:33,000 --> 00:26:43,000
the properties of the threat in terms of, you know, where where they operate, what entities, you know, what code they're touching, things like that

168
00:26:44,000 --> 00:26:54,000
. I mentioned graph. So you can you can model the system as a graph. You can model the attacks as pathways through the graph, and that gives you a rich set of algorithms and queries

169
00:26:54,000 --> 00:27:03,000
. You can cluster things you can look at, you know, cycles. There's a lot of rich mathematics around modeling. As a graph, you can have rules. So that's kind of where

170
00:27:03,000 --> 00:27:13,000
we're going to be talking. Um, you know what can be rules of thumb that can be based on the score research that can be based on experience? And I think, you know, the the emerging

171
00:27:13,000 --> 00:27:23,000
future approach is to use A.I. with a combination of all of those things. And help, of course, has some machine learning underneath it. But you can use

172

00:27:23,000 --> 00:27:33,000
AI to predict these relationships to model these statistics and provide a predictive model for where things are trending, even generating

173
00:27:33,000 --> 00:27:42,000
graph relationships to investigate. Or, as I said, clustering things or finding new powers, setting weights, tagging things

174
00:27:42,000 --> 00:27:52,000
, we'll talk about tagging. So I think it's really in the next five years going to be an important component. It has its own considerations around risks and bias and

175
00:27:52,000 --> 00:27:57,000
training and all these things. But I think it will be important

176
00:27:57,000 --> 00:28:07,000
. And then again, back to that get ops model, you know, we really need to embrace this notion that we're not approving by just consent

177
00:28:07,000 --> 00:28:17,000
versus documentation or consensus review or lack of any objection. We're really, you know, enforcing the process through some

178
00:28:17,000 --> 00:28:27,000
artifact and some, you know, PR a pull request. If you're familiar with get GitHub, you're you're noting the approval and all the

179
00:28:27,000 --> 00:28:37,000
criteria for that approval. Right? And then CSP is in three parts play an important part in this. You know, they'll have to maintain IOSCO for the components

180
00:28:37,000 --> 00:28:46,000
and the control implementation details the tests and you want to be able to have kind of a dry run. This is one thing that you can do in a system like Kubernetes. You have policies

181
00:28:46,000 --> 00:28:56,000
, code and checking configurations. You can set out all your rules in

something like open policy agent and all your configurations in the declared state.

182
00:28:56,000 --> 00:29:06,000
And then you can do a dry run and say, you know, does everything satisfy the rules before you deploy it? And then, of course, you can have your runtime inventory test and there you need

183
00:29:05,000 --> 00:29:15,000
to detect drift. So if I say I'm declaring a particular configuration and deploy that, I need to have some some mechanism to say, has it been tampered with by you?

184
00:29:15,000 --> 00:29:25,000
know, internal or external actors? And all of this allows for coverage metrics. And I want those coverage metrics that at all levels, I want that at the at the

185
00:29:25,000 --> 00:29:34,000
, you know, catalog control level. How we covered everything at the risk model, which will show shortly are the risks defined or they cover it or they implement it

186
00:29:34,000 --> 00:29:44,000
. And you know, from a code perspective, how much code am I introducing each stop and are the approvals? Are the test matching up

187
00:29:44,000 --> 00:29:53,000
? So let's let's talk about this rules engine. So again, we've talked about having desired state. We're going to have rules about that use

188
00:29:53,000 --> 00:30:03,000
and reason over the catalogs that controls the profiles. We're going to have what I call declared state the, you know, the component configurations and the security

189
00:30:03,000 --> 00:30:13,000
capability configurations. We're going to have some understanding of threats and all of this so that we can reinforce those control implementations with threat data. We're going to have

190

00:30:13,000 --> 00:30:22,000
code tests, rules that test things that's going to be directly encoded and things like the SAP. And then that's going to generate from

191
00:30:22,000 --> 00:30:32,000
the inventory of the song. And let's say that helps reinforce this kind of lifecycle. You know,

192
00:30:32,000 --> 00:30:42,000
continuous updates. So we're going to have controls updated by catalogs and agencies and the regulatory folks. We're going to have the component that's

193
00:30:42,000 --> 00:30:51,000
needing to be updated. We're going to have this kind of threat based model of the system. And then that's going to inform our testing and

194
00:30:51,000 --> 00:31:01,000
as I mentioned, there is some AI work being done in the open source community. So you should definitely check out these projects. I think that's where things are going. So

195
00:31:01,000 --> 00:31:11,000
decision gates, for those people to talk about that, we're working on some some research around kind of building a time machine of deaths for the entire system

196
00:31:11,000 --> 00:31:21,000
. And then, you know, remediation, there's a lot of commercial and open source work about automating the remediation and generating actual

197
00:31:21,000 --> 00:31:30,000
code changes based on findings and programs. So as those who have done better, have you know that you need to

198
00:31:30,000 --> 00:31:40,000
do online? You know that you need to report where things are on a monthly basis, but you're really kind of trying to move towards a continuous dev ops model. So you really want to

199

00:31:40,000 --> 00:31:49,000
make sure that all of these things are being updated all of the time and the changes are propagating automatically through the pipelines as long as they're guided by those PR approvals.

200
00:31:49,000 --> 00:31:59,000
And so I think somewhere I may have lost my mind reminders to do the demo. So I think I'll jump into the demo

201
00:31:59,000 --> 00:32:09,000
before we talk about the more graphic stuff. So to do that, I'm going to jump over to my Ricoh playground. So as

202
00:32:09,000 --> 00:32:19,000
I mentioned, Riga is a language for declaring rules and then validating those rules very effectively and very efficiently. But before you can do any validation, you have to have some

203
00:32:19,000 --> 00:32:28,000
data. So this is a very simplified version of inputs you might get from a catalog. And so you're going to define your controls. They're going to have statements

204
00:32:29,000 --> 00:32:39,000
, you have properties, those who have worked with Moscow now at the Level two three, and this will look familiar for those who have lost yet. It's it's just data

205
00:32:38,000 --> 00:32:48,000
. And so you're looking at what controls as the system needs to employ on top of kind of the controlled

206
00:32:48,000 --> 00:32:58,000
data, the the system will have some some capability, some security capabilities to protect, detect or

207
00:32:58,000 --> 00:33:08,000
respond when that kind of Gov. Kamado. So here we've got some inputs and shares the CCP's capabilities for a particular component

208
00:33:08,000 --> 00:33:17,000

and allows us to reason over different types of protection that might provide. And you know what it implements

209
00:33:17,000 --> 00:33:27,000
? And then, you know, for for noting that there is discussion around creating a rules or test schema in Moscow. It's not official yet.

210
00:33:27,000 --> 00:33:37,000
So some of this is his prototype. But I'm fairly confident that the folks on this call participating in that we will get to some implementation

211
00:33:37,000 --> 00:33:46,000
of rules in the not too distant future. And so again, as a CSP, you're typically providing components. Your component

212
00:33:46,000 --> 00:33:56,000
abstract class will have to be implemented and we'll have to provide evidence of these security capabilities. So now we get to the point we're actually

213
00:33:56,000 --> 00:34:06,000
playing with the the policy. So in the Riga language, you know, the syntax. If you've ever worked with data log, it's it's very similar to data log

214
00:34:05,000 --> 00:34:15,000
, but these are essentially, you know, tools that the the engine will interpret and tell you if everything in these rules sets are true,

215
00:34:16,000 --> 00:34:25,000
And so this it's not it's not like Java or C code was executing this and doing things with, you know, branching logic

216
00:34:25,000 --> 00:34:35,000
is basically saying is every single thing I'm certain here true. And if it is, then the output of that should reflect that. And as I mentioned,

217
00:34:35,000 --> 00:34:44,000
today, and so like a Kubernetes system, open policy agent is used

really to make very binary decisions. And if you're smart communities, you can use it for

218
00:34:44,000 --> 00:34:50,000
admission control and say, should I put things new things into the system or not?

219
00:34:50,000 --> 00:35:00,000
hear a side effect of where you go and open policy agent is, you can actually generate code. So here we've we've mentioned there were basing

220
00:35:00,000 --> 00:35:10,000
things on on threats. So we've got threats identified in our capabilities, you know, the threats that we're going to protect against

221
00:35:09,000 --> 00:35:19,000
and we need to find some mitigation execution of unauthorized container. And now we're going to evaluate and. And so it's showing you that it's

222
00:35:20,000 --> 00:35:29,000
, you know, everything in green is true, but as a side effect, it's actually generated in the output is here's a score based on weightings that we provide right

223
00:35:29,000 --> 00:35:39,000
? And I'll highlight, you know, you might have in the Gupkar methodology, you know, a heat map value based on that miter attack or other threat model and

224
00:35:38,000 --> 00:35:49,000
how the the controls intersect with those kind of those threats. And you can see that it's loaded up different weights

225
00:35:49,000 --> 00:35:59,000
. And so here you can see that in addition to saying that, everything that I've declared in my rules is true, that I am interested

226
00:35:59,000 --> 00:36:09,000

as a subject matter expert in this particular threat. It's also calculating the score, you know, weighted or otherwise of how this particular example is covering

227
00:36:09,000 --> 00:36:19,000
and protecting. So you get that protection about your score and you get that coverage score. So beyond just saying now have a how's my

228
00:36:19,000 --> 00:36:28,000
component definition, my abstract class, my security capability from the CSP, again, in my abstract definition of what I plan to provide

229
00:36:28,000 --> 00:36:38,000
, is that really doing what I needed to do? You're going to you're going to subject matter expert is going to define some rules, right? So we're going to say that, you know, I'm interested in containers

230
00:36:37,000 --> 00:36:47,000
. I needed to have a signature capability and to be able to sign digitally sign any container that's going to be allowed into the system and

231
00:36:48,000 --> 00:36:57,000
in doing that, I'm going to want to evaluate that with some methodology doesn't mean this is using a particular query syntax, but it could be any any

232
00:36:58,000 --> 00:37:07,000
kind of evaluation syntax you're interested in. I want to find containers with signatures that are undefined, and that means that it would fail, right? So if

233
00:37:07,000 --> 00:37:17,000
I'm a lot of the concern is that unauthorized containers are going to be allowed into my system, my security capabilities to prevent that, my test so that

234
00:37:17,000 --> 00:37:27,000
at the inventory level. Am I going to be able to detect that container without a signature? And again, as a side

235

00:37:27,000 --> 00:37:36,000
effect, you'll see here that it will actually generate specific rule implementations. So we haven't gotten yet to a running system

236
00:37:36,000 --> 00:37:46,000
. But we're saying that in in my analysis and in my south planning. And now I'm tying this down to a particular

237
00:37:46,000 --> 00:37:56,000
implementation. I'm going to be looking for containers with signature value undefined. And again, it's covering, you know, it's mapped to all the threats and I'm interested in. And then

238
00:37:56,000 --> 00:38:06,000
what are we going to do with those rule implementations? We're going to tie those two control implementations so that I think the fundamental shift in mentality here

239
00:38:06,000 --> 00:38:15,000
is you're letting the rules engine match things up and define if the implementation is correct, either be a coverage or

240
00:38:15,000 --> 00:38:25,000
threat scoring or both. And so when we evaluate the control implementation scoring, I'm saying, you know, I've got to have a particular protection threshold

241
00:38:25,000 --> 00:38:35,000
old. I've got to have a particular coverage threshold, you know, I'm interested and I want to make sure that those rules are associated with it. What's getting generated is

242
00:38:35,000 --> 00:38:45,000
the ask out for, you know, pseudo issue of scale in this demo for control implementation for component definition. So here it's

243
00:38:45,000 --> 00:38:55,000
tied it all together. It's, you know, generated the implement requirements based on the control. I'd use reference from the catalog. It's then connected. Those

244

00:38:55,000 --> 00:39:05,000
to the conditions and rules for gathering evidence for that. And again, it's, you know, adhering to the idea semantics

245
00:39:05,000 --> 00:39:15,000
and linkages in Moscow. And then in terms of testing all this and, you know, collaborating with both the CSP

246
00:39:15,000 --> 00:39:24,000
and the three power unit and generate an assessment plan. And so that's really just binding those rules that are kind of abstract to actual inventory

247
00:39:24,000 --> 00:39:34,000
tasks and so on. Doing that now you get an assessment plan that is connected to the SSP and you can see

248
00:39:35,000 --> 00:39:44,000
, you know, very specific set of tasks. You have your control implementations. And so this would be severe. So then once you have a running

249
00:39:44,000 --> 00:39:54,000
system, you can use those queries connected. You know, they're fully traceable and connected to all the controls, controls, limitations when you gather that evidence. You know, now you've

250
00:39:53,000 --> 00:40:03,000
you have confidence that the system is implemented correctly. So I'm going checking time now, getting closer

251
00:40:04,000 --> 00:40:13,000
have a bit more to cover, but I think we'll have to move a little quickly. And so I talked about all those relationships earlier. You know, I

252
00:40:13,000 --> 00:40:23,000
think in graphs, I think there was a Microsoft quote, you know, the attacker's thinking, gosh and defenders thinking lists, and that's bad. So, you know, we try to think and draft. So if you

253

00:40:23,000 --> 00:40:33,000
think about what isn't necessary, it's kind of a view of the system, but it's a graph of the controls and the components and how they're implemented. Work together, how to run time

254
00:40:33,000 --> 00:40:42,000
inventory level. You can, you know, encode threats and and relationships, score those relationships before you can then run queries

255
00:40:42,000 --> 00:40:52,000
over those draft models and you can find attack paths so that you can inform your rules doing and then you find where things are clustered together. So you

256
00:40:52,000 --> 00:41:02,000
can inform your catalogs and your control implementations. And so this is kind of here's an example of a real system commercially, but I

257
00:41:02,000 --> 00:41:11,000
I do reference the open source free version and the summary. So this is just a graph view of a system

258
00:41:11,000 --> 00:41:21,000
where, you know, you notice where we're encoding and enriching these relationships with specific threat identifiers and metrics. So

259
00:41:21,000 --> 00:41:31,000
that you really then can start to reason over attack pathways and security capabilities, both as an agency, as a three power. And certainly

260
00:41:31,000 --> 00:41:41,000
as a CSP. If you're if you're designing a system, I mentioned that, you know, entity extraction, semantic search as ML, it's being used today. I think a number of

261
00:41:40,000 --> 00:41:46,000
folks on the call had presented some information about that or open source. You know, there are

262

00:41:46,000 --> 00:41:51,000
different models being used all Open-Source that are helpful

263
00:41:51,000 --> 00:42:01,000
. Again, if you're familiar with systems like Neo4j, which again is open source and commercial, but Neo4j comes with a lot of rich graph ml capabilities

264
00:42:02,000 --> 00:42:11,000
, so you can construct a lot of these models from those libraries. You don't have to invent these yourselves, and then you can start to reason about, you know, is my speeds

265
00:42:11,000 --> 00:42:21,000
covering everything I need to cover is my is my three power covering everything within a particular control test or in their sampling methodology. And then what is my overall

266
00:42:21,000 --> 00:42:31,000
threat exposure which tying it all back to? I think what the PMO has stated. We want to get some more that risk management mentality. And I think that's, you know, we're graph south a lot and the amount

267
00:42:31,000 --> 00:42:41,000
we're constantly learning or tying down all these threat models to specific queries. And that's information we run in in both federal environments.

268
00:42:41,000 --> 00:42:50,000
and commercial environments. We run all of our scenario planning or incident response for disaster recovery for business continuity. Through that graph of you

269
00:42:50,000 --> 00:43:00,000
, I showed previously and now are reasoning about the system, not a very linear way from the document. The checklists do you have those as artifacts

270
00:43:00,000 --> 00:43:10,000
, but we're generating those from graph queries and graph reasoning. So they're releasing in the context of the system and

```
271
00:43:10,000 --> 00:43:20,000
all the different attack failure pathways. So to sum up, I just want
to leave a few some time for questions and

272
00:43:19,000 --> 00:43:29,000
you know, we look at the world from an agency perspective as ASCO
enabling compliance as code generating artifacts that are usable

273
00:43:30,000 --> 00:43:39,000
, that are verifiable as they come in, that are testable. We we want
to enable that continuous DevSecOps continuous data

274
00:43:39,000 --> 00:43:49,000
model so that you can detect drift those alerts and then really report
on precise coverage metrics. And then ideally, we'd

275
00:43:49,000 --> 00:43:59,000
like to see that kind of PR issue, right? So we have approval gates
encode in pipelines and so that we start to think of the system as

276
00:43:59,000 --> 00:44:08,000
developers do that there's no, you know, there's no sacred cows,
there's no there's still no cattle in and what you're building that
it's all reconstructed

277
00:44:08,000 --> 00:44:18,000
from code and rules and allows us to definitively show compliance. And
at any point in time,

278
00:44:18,000 --> 00:44:28,000
no matter what change, and do it very quickly and effectively with
minimal human rework. And so I'll just cover this very quickly

279
00:44:28,000 --> 00:44:38,000
. You know, we need we need more support for automation. So those of
us in science and related vendors who are contributing contributing to
open source

280
00:44:38,000 --> 00:44:48,000
are trying to help. I think we need I think one of the presenters
```

earlier had mentioned their implementation of an API. I think that's the right approach. We need more API driven interaction

281
00:44:48,000 --> 00:44:58,000
and eventually we'll need ML models that are, you know, well, synchronized, open, transparent. We'll need, you know, artifacts contributed

282
00:44:58,000 --> 00:45:08,000
catalogs, components. And so, you know, folks like defense unicorns have helped create a lot of great scale examples and catalogs for things like Iron Bank and such

283
00:45:08,000 --> 00:45:18,000
. We need more open source definitions, and we need the vendors and ISPs to step up and and see if we're helping that we're going to be running through all seen south projects

284
00:45:18,000 --> 00:45:28,000
that support communities and helping them define off scale for all their components. We need GRC, folks. If you're on the call, please, please

285
00:45:28,000 --> 00:45:37,000
help us bonanza word and excel. Copy paste your tools need to support off scale both ingress and egress. Three powers

286
00:45:37,000 --> 00:45:47,000
, please. We know you have accreditation requirements, but we want to help you use our scale. And so the system is self documenting and self testing

287
00:45:47,000 --> 00:45:57,000
, and that you, as subject matter experts, help inform that role building. And then again, CSP is to support this effort at every stop. Agency leadership

288
00:45:56,000 --> 00:46:06,000
, and they need to embrace it. And I think we've talked about some of the benefits to their hiring and retention. If they do, yeah, so there we are. We

289
00:46:06,000 --> 00:46:16,000
we will put everything this presentation, all the example Rigo code, the graph queries will eventually be we had we had our KubeCon conference

290
00:46:16,000 --> 00:46:26,000
last week, so I got a bit behind in getting everything on the record. But it will be there the repo for anyone who's interested in not the graph

291
00:46:27,000 --> 00:46:36,000
open source graph engine that we use is called Starbase that anyone who has with does take a look. And then again, we're happy to support the community

292
00:46:36,000 --> 00:46:46,000
. Please, you know, reach out, schedule a call with us. We're happy to spend time with agencies. Three powers even see ISPs for sure

293
00:46:46,000 --> 00:46:55,000
. Let us know how we can help you on your Oscar journey and apply some of these code templates and examples in your specific use. So with that, I

294
00:46:56,000 --> 00:47:05,000
hopefully have enough time for questions, but I'll go ahead and stop there. Thank you very much, Robert, so we can stop the recording right now and open the floor