package com.twitter.visibility.rules

import com.twitter.abdecider.LoggingABDecider

import com.twitter.timelines.configapi.Params

import com.twitter.visibility.configapi.configs.DeciderKey

import com.twitter.visibility.configapi.params.RuleParam

import com.twitter.visibility.configapi.params.RuleParams

import com.twitter.visibility.configapi.params.RuleParams.\_

import com.twitter.visibility.features.Feature

import com.twitter.visibility.models.UserLabelValue

import com.twitter.visibility.models.UserLabelValue.\_

import com.twitter.visibility.rules.Condition.\_

import com.twitter.visibility.rules.Reason.\_

import com.twitter.visibility.rules.RuleActionSourceBuilder.UserSafetyLabelSourceBuilder

object AbusiveRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

Abusive

)

object DoNotAmplifyUserRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

DoNotAmplify

)

object AbusiveHighRecallRule

extends AuthorLabelAndNonFollowerViewerRule(

Drop(Unspecified),

AbusiveHighRecall

)

object CompromisedRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

Compromised

)

object DuplicateContentRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

DuplicateContent

)

object EngagementSpammerRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

EngagementSpammer

)

object EngagementSpammerHighRecallRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

EngagementSpammerHighRecall

)

object LiveLowQualityRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

LiveLowQuality

)

object LowQualityRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

LowQuality

)

object LowQualityHighRecallRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

LowQualityHighRecall

)

object NotGraduatedRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NotGraduated

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableNotGraduatedDropRuleParam)

override def holdbacks: Seq[RuleParam[Boolean]] = Seq(

NotGraduatedUserLabelRuleHoldbackExperimentParam)

}

abstract class BaseNsfwHighPrecisionRule()

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

UserLabelValue.NsfwHighPrecision

)

object NsfwHighPrecisionRule

extends BaseNsfwHighPrecisionRule()

object NsfwHighRecallRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwHighRecall

)

abstract class BaseNsfwNearPerfectAuthorRule()

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwNearPerfect

)

object NsfwNearPerfectAuthorRule extends BaseNsfwNearPerfectAuthorRule()

object NsfwAvatarImageRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwAvatarImage

)

object NsfwBannerImageRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwBannerImage

)

object NsfwSensitiveRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwSensitive

)

object ReadOnlyRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

ReadOnly

)

object RecommendationsBlacklistRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

RecommendationsBlacklist

)

sealed abstract class BaseSpamHighRecallRule(val holdback: RuleParam[Boolean])

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

SpamHighRecall

) {

override val holdbacks: Seq[RuleParam[Boolean]] = Seq(holdback)

}

object SpamHighRecallRule extends BaseSpamHighRecallRule(RuleParams.False)

object DeciderableSpamHighRecallRule extends BaseSpamHighRecallRule(RuleParams.False)

object SearchBlacklistRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

SearchBlacklist

)

object SearchNsfwTextRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwText

) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableNsfwTextSectioningRuleParam)

}

object SpammyFollowerRule

extends OnlyWhenNotAuthorViewerRule(

Drop(Unspecified),

And(

Or(

AuthorHasLabel(Compromised),

AuthorHasLabel(EngagementSpammer),

AuthorHasLabel(EngagementSpammerHighRecall),

AuthorHasLabel(LowQuality),

AuthorHasLabel(ReadOnly),

AuthorHasLabel(SpamHighRecall)

),

Or(

LoggedOutViewer,

And(

NonAuthorViewer,

ViewerHasUqfEnabled,

Or(

And(

ProtectedViewer,

LoggedOutOrViewerNotFollowingAuthor,

Not(AuthorDoesFollowViewer)

),

And(Not(ProtectedViewer), LoggedOutOrViewerNotFollowingAuthor)

)

)

)

)

)

abstract class NonFollowerWithUqfUserLabelDropRule(labelValue: UserLabelValue)

extends ConditionWithUserLabelRule(

Drop(Unspecified),

And(

Or(

LoggedOutViewer,

And(Not(ViewerDoesFollowAuthor), ViewerHasUqfEnabled)

)

),

labelValue

)

object EngagementSpammerNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(

EngagementSpammer

)

object EngagementSpammerHighRecallNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(

EngagementSpammerHighRecall

)

object SpamHighRecallNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(

SpamHighRecall

)

object CompromisedNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(

Compromised

)

object ReadOnlyNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(

ReadOnly

)

object LowQualityNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(

LowQuality

)

object TsViolationRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

TsViolation

)

object DownrankSpamReplyAllViewersRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

DownrankSpamReply

) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableDownrankSpamReplySectioningRuleParam)

}

object DownrankSpamReplyNonAuthorRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

DownrankSpamReply

) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableDownrankSpamReplySectioningRuleParam)

}

object DownrankSpamReplyNonFollowerWithUqfRule

extends NonFollowerWithUqfUserLabelDropRule(DownrankSpamReply) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableDownrankSpamReplySectioningRuleParam)

}

object NsfwTextAllUsersDropRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

NsfwText

) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableNsfwTextSectioningRuleParam)

}

object NsfwTextNonAuthorDropRule

extends WhenAuthorUserLabelPresentRule(

Drop(Unspecified),

DownrankSpamReply

) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(EnableNsfwTextSectioningRuleParam)

}

abstract class DeciderableSpamHighRecallAuthorLabelRule(action: Action)

extends RuleWithConstantAction(

action,

And(

NonAuthorViewer,

SelfReply,

AuthorHasLabel(SpamHighRecall, shortCircuitable = false)

)

) {

override def preFilter(

evaluationContext: EvaluationContext,

featureMap: Map[Feature[\_], Any],

abDecider: LoggingABDecider

): PreFilterResult = {

Filtered

}

}

object DeciderableSpamHighRecallAuthorLabelDropRule

extends DeciderableSpamHighRecallAuthorLabelRule(Drop(Unspecified))

object DeciderableSpamHighRecallAuthorLabelTombstoneRule

extends DeciderableSpamHighRecallAuthorLabelRule(Tombstone(Epitaph.Unavailable))

object DoNotAmplifyNonFollowerRule

extends AuthorLabelAndNonFollowerViewerRule(

Drop(Unspecified),

DoNotAmplify

)

object NotGraduatedNonFollowerRule

extends AuthorLabelAndNonFollowerViewerRule(

Drop(Unspecified),

NotGraduated

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableNotGraduatedDropRuleParam)

override def holdbacks: Seq[RuleParam[Boolean]] = Seq(

NotGraduatedUserLabelRuleHoldbackExperimentParam)

}

object DoNotAmplifySectionUserRule

extends AuthorLabelWithNotInnerCircleOfFriendsRule(

ConversationSectionAbusiveQuality,

DoNotAmplify)

with DoesLogVerdictDecidered {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

UserSafetyLabelSourceBuilder(DoNotAmplify))

override def verdictLogDeciderKey = DeciderKey.EnableDownlevelRuleVerdictLogging

}

object SpammyUserModelHighPrecisionDropTweetRule

extends AuthorLabelAndNonFollowerViewerRule(

Drop(Unspecified),

SpammyUserModelHighPrecision,

)

with DoesLogVerdictDecidered {

override def isEnabled(params: Params): Boolean =

params(EnableSpammyUserModelTweetDropRuleParam)

override def verdictLogDeciderKey: DeciderKey.Value =

DeciderKey.EnableSpammyTweetRuleVerdictLogging

}

object LikelyIvsLabelNonFollowerDropUserRule extends LikelyIvsLabelNonFollowerDropRule

object SearchLikelyIvsLabelNonFollowerDropUserRule extends LikelyIvsLabelNonFollowerDropRule

object NsfwHighPrecisionUserLabelAvoidTweetRule

extends UserHasLabelRule(

Avoid(),

UserLabelValue.NsfwHighPrecision

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

NsfwHighPrecisionUserLabelAvoidTweetRuleEnabledParam)

}