1)С текущей формулой расчета множителя мне не удалось снизить урон ровно в три раза, множитель выходит равным ~0.3333246692727997, не думаю, что стоит из-за такой неточности перерабатывать формулу расчета или добавлять новый функционал.

Не уверен, что необходимо добавлять иконку, ведь согласно тз этот перк действует на ровно одну атаку, у него нет времени действия, но поскольку в качестве примера был указан перк **TOUGH\_BODY** - сделал по аналогии.

2)Можно было бы создать отдельный триггер для обработки удара с оружием, но это решение показалось мне слишком объемным, поэтому решил реализовать через **OR()** в условии, существует вариант вынести в отдельный **AND()** проверку **event.Target=="Enemy"** ради устранения дублирования, но это породило бы вот такую конструкцию:

**AND (event.Target=="Enemy", OR (AND ( event.Animation=="Unarmed", random() < attackBoostChanceUnarmed ), AND ( event.Animation=="Weapon", random() < attackBoostChanceArmed ))),** а это уже совсем тяжело для восприятия.

3)По третьему заданию комментариев не возникло.

4) Если я правильно понимаю механику работы функции **random(),** то она каждый вызов отдает новое случайное число. Из этого следует проблема, например в триггере **RAGE\_CHECK\_SUCCESS** **random()** отдала 0.4, то есть событие **Success** создано не будет, далее срабатывает **RAGE\_CHECK\_FAIL**, который по логике должен создать событие **Fail**, но поскольку функция **random()** каждый раз отдает новое число, то с некоторой вероятностью в **condition** триггера **RAGE\_CHECK\_FAIL** мы можем получить число, к примеру, 0.1. Отсюда следует, что ни одно из событий не будет создано, что противоречит логике перка.

Также вероятна и обратная ситуация: событие **Success** было создано, а потом **random()** в **RAGE\_CHECK\_FAIL** отдала 0.4, получается, что создаются оба события. Думаю, это можно решить путем добавления **ModFlag** при **Success** на 1 кадр и уже в зависимости от существования этого мода создавать/не создавать событие **Fail**, таким образом мы избавимся от второго вызова **random()** в **RAGE\_CHECK\_FAIL**.

5)При попадании по игроку, после расчета урона от удара, с вероятностью 3/10 враг получает мод, снижающий урон в 2 раза на 5 секунд. Пока мод действует, его новый экземпляр не может быть создан, то есть, у эффекта нет возможности суммироваться.

Постановка задачи звучит довольно размыто. "Сделайте шанс срабатывания пропорциональным урону" – может подразумевать несколько разных вариантов, но любой из них требует, как минимум, свойства **damage** для события **PostHit**, где будет содержаться рассчитанный урон от удара. Вот пара вариантов решения, пришедших мне в голову:

1. Я не знаю, какого порядка цифры урона обычно, но представим, что от 1 до 1000, меняем **chance=0.3** на **chanceFactor = 0.0003**, а в условии **random() < chance** на **random() < chanceFactor \* Event.Damage**, чем выше урон, тем выше вероятность срабатывания, пропорция соблюдена.

2. Еще напрашивается вариант с уроном относительно максимального здоровья игрока, для этого необходимо вместо **PostHit.Damage** передать свойство **PostHit.RelatedDamage**, которое будет содержать в себе отношение рассчитанного урона от удара к максимальному здоровью игрока.

В таком случае в условии меняем **random() < chance** на **random() < event.RelatedDamage**. Чем больше отношение урона к максимальному здоровью, тем больше вероятность срабатывания.