WELCOME TO ANNUAL TECHNICAL HEALTH REVIEW FOR

FORWARD LOOKING STATEMENTS

So these are the forward looking statements which salesforce has done over the years. One thing to be noted is kindly do not take any business decisions on the findings we have got till now.

AGENDA:

So the agendas of today's call divides into four parts

first the overview: what you are trying to achieve through this investigation and through this presentation

second is CODE SCAN health review that you have done on your developer org and performance review that you have done on your production site and next steps on how to fix the issues and prioritize, and how to strategize everything that you have told.

OVERVIEW

So in the overview, this presentation comprises generally of two parts: first is code scan and 2nd is performance review. In the code scan which we have done, we have got 167 severe vulnerabilities found.  
We would suggest you to avoid calling SOQL and DML inside the loops. We will get into this in detail in the further slides. This is just an overview of what we are going to cover in this presentation.

and the second part of this is, we have performance review where we do some of our analysis on the production to see which are slow running Apex classes, VF pages and bifurcation of various classes and what governor limits were met from your end in last 30 days.

Code Scan Health Review:

so now lets start to speak abut the code scan health review. When this ATHR started what we do? we ask our customers to give us a freshly prepared developer sandbox which has the all the latest code from production, and all the code scan we run are based on some of the principles which we have developed or have found based on number of customers we have dealt with so far.

BREAKDOWN ISSUES FOUND

before I get into the statistics, I would like to discuss what different severity means.  
so in front of you, you can see there are 5 major categories which we have been distiguished based on the issues.   
first is blocker: blocker generally comprises of the issues which might destabilize the functionality.   
For example any issues which might cause a null pointer exception or a governor limit  
-for example adding an SOQL query inside a for loop.  
  
Critical: critical generally comprises of security issues, or performance issues. These are such issues which are not blocker right now, but depending on future data, it may become a critical issue.  
For example hard coded salesforce IP, hard coded salesforce ID.

Major will generally comprises of other connectivity issues, performance issues and insecure applications.   
for example not rising,NOT CONFIGURING with sharing, without sharing ON CLASSES WHICH HAVE DML IMPACT.

Minor and Info are the 2 categories which deals with the violation of best practices.  
Information generally comprise of the poor build quality .

both are about best practices, but finally major is of a higher priority and information is of lower priority.  
so uh in case of blockers we have found 167 blockers on your org. 154 critical 1422 major 2483 minor issues and around 5600 are informational issues total issues accumulating to pay around 10000 issues.

SOQL INSIDE LOOPS:  
This is an issue which we tend to see with many customers, when queries are placed inside a for loop, a query is executed on each iteration and the governor limit will be reached easily.  
For this we would suggest you to move the database operations outside of for loops.

HEALTH REVIEW ANALYSIS-  
so before I get into statistics again, you can see there is a distinguish between two factors one is with test and without test. So what does it mean? so, when we retrieve the meta data from the developer sandbox we retrieve the metadata, that constitutes everything: your apex class, triggers, trigger handler, helper classes your aura controllers everything we retrieve. That also entails the test classes. so first when you runout code scan analysis we do everything but, what is the necessity of doing without test classes, that is only with functionality classes is, there are many kind of issues, for example, writing a single query inside a for loop, if that is done inside a functionality class, there's a high probability that it will run into a governor limit issue-affecting the entire functionality.

but if I do it in a test class it might not cause an issue because you would be using data factory and in data factory, you might be just creating two or three records which will not create damage, so that's why we have done this bifurcation. so that you guys know that even in blockers or critical or major or minor or info you could understand what to focus on. so if you have to take the example of say critical I can see with-test critical count is 154 without test is 39 that means that there are 154 issues in your test classes and 39 issues there are there in functionality classes so this tells you the differentiation that how many issues are there your functionality classes and how many issues are there in test class.

HEALTH REVIEW ANALYSIS FOR IMAPCTING VULNERABILITIES

we have come to high impacting vulnerabilities and by impacting vulnerabilities I mean blocker issues.

so there are three major types of vulnerabilities which are have found to be in your org:  
first is avoid calling dml inside the loop, this is not recommended by Salesforce because this tends to create a governor limitations which will in turn stop the functionality   
avoid calling soql without where clauss or limit clauss . it's always recommended to use filters in your query so that it results in index search instead of table scan which will make your entire code slow.

broken null check : what happens is generally, what we see is, developers tend to write the null checks in such a way that they themselves tend to go through null pointer exception.  
for example if you incorrectly use AND and OR operators in null check, so those things go through null pointer exception. so this is also one of the issues that we found in your org

HEALTH REVIEW ANALYSIS FOR CRITICAL vulnerabilities

with this we come to critical vulnerabilities in critical vulnerabilities we see the top one to be avoided using @isTest this is something we don't recommend uh our customers to use it because what happens that if we use this, when you tend to push this code to production and you run the test class it tends to fail, because the data is not available in production.   
Also, Avoid using hardcoded IP we don't recommend using hardcoded IP because again that is a very bad practice and in future if those things change, you need to make a lot of customized and manual changes and avoid using hardcoded Salesforce ID this is also not recommended because if you get to use hardcoded sales force id when you put the code from 1 org to another org, the ID's change that will again result in failure of your functionality

HEALTH REVIEW ANALYSIS FOR MAJOR VULNERABILITIES-

coming to major vulnerabilities again the first one to be avoid calling negative expression.  
we don't recommend using a negative expression because they tend to do table scan again, instead of index scan, and make the query ineffective.  
Avoid using nested if statement, this will result in too much CPU consumption.  
Avoid running Test.isRunningTest() as well.  
Avoid deeply nested if statement, by deeply we meen more than 2, If you use deeply nested if statement, it is going to consume more CPU time and make your code more slower.  
  
CODE SCAN FILE DELIVERABLES

one more thing I would like to show you, i've been giving such an overview, you might ask me ‘where are the issues?’ so along with this will be providing you with this excel sheet, this excel sheet will contain all the issues by your class and line numbering what issues you get, what issue is there on that particular line number, and you can simply go on those classes, go to line number and you could fix it up. so this presentation is just an overview and you'll get all the details in that excel sheet.

Information On Other Issues

these are the help particles which are corresponding to what we discussed till now, so kindly go through this and see if something can be fixed up.

KEY RECOMMENDATIONS

what are the key recommendations? so firstly we're talking about injection vulnerability and app logic vulnerability. This is not something very specific to your org.  
This is something we tell all of our customers to look into because our sales force is growing, threat to it is also growing so it's always good to help your the developers have hands on this.

now coming to the second part: how to look into this, so first and foremost the priority goes to blocker. blocker, critical, major, these three are the primary, then comes info and minor.   
Info you can ignore for the time being because those are more like more quality issues.

when we will be giving you the excel sheet, the excel sheet should be reviewed as a punchlist because I may tell you to use limit cause and you are saying ‘no’- our business does not acquire, so we can raise the possible false positives. so first review with the developer, to see why this code is written like this, if it is something that can be fixed.

When you're fixing your lower environment, do a full UAT testing and then push it to production

------------------------------------------------------------------------------------------------------------------------------------------

PERFORMANCE

we'll come to the second part which is performance analysis.  
so now in performance analysis, according to GDPR rule, we just intend to be get 30 days of the data, that is from 12 September 12th OCT will be used for analysing your production org.

so in performance analysis what we do is, we tend to analyze the slow running VF pages slow running apex classes, and the bifurcation, that is what are the issues in those classes

PERFORMANCE: VF PAGES

So we have seen a few slow running VF pages in last 30 days, which is to be considered  
  
PERFORMANCE: VF PAGES

coming to the second part-the apex classes, before I get into this I would like to discuss two factors that you can see-The two columns which are ‘all transactions’ and transaction with runtime greater than 5 second, and something called percentage of slow loads. So what happens is, generally ,whenever a transaction is running more than five seconds we tend to consider it as slow. there are various internal factors that come into play, be it restrictions, be it concurrency , many factors for which 5 second is the benchmark which has come to our assessment

so in this as you can see all transactions that means that from 28 Aug 2022 – 28 september 2022, /apex/quessionaireResponse ran 90 times, among that there were 37 times when it took more than 5 seconds to load. So (37/90)\*100 is 41.1. This is how the calculations are done here. And In your case, these 3 entry points have taken a lot of time to load.

[These are synchronous transactions and API calls are not included in these]

PERFORMANCE OF APEX:

Details of bifurcation:  
Let me say, We have given entry point and the runtime is high,  
that runtime is divided into 3 parts-Logic time, DB Time and Callout time

If logic time is high, it means the logic is to be fixed.  
If DB time is high, there are too many SOQL queries happening.  
If the call out time is high, it means the end points need to fixed.

We’ll be giving the excel sheet and it contains a lot of data about it, whichever you feel like fixing, can be fixed using that sheet.

APEX GOVERNOR LIMITS HIT  
This slide shows about the entry point that hit the governor limit within the last 30 days.  
That is ‘ ‘ with a count of ‘ ‘.

KEY Recommendations

Performance Best Practices

We encourage you to familiarize yourself with these performance profiling tools and resources. In this guide, you can learn about both the Force.com performance profiling tools and their associated methodologies, which have been informed by recommendations from salesforce.com Customer Support and our own engineering teams.

What customers are expecting?  
How does dynamic scan be more affective than static scan  
bifurcation  
db/cpu/callout

Per Transaction limit,  
call to action for the customer-strategy