|  |
| --- |
| Title : Parsing Defects/Parsing Improvements: Where To Start When You Don’t Know |
| URL Name |
| **PART V of VI: Parsing Improvement: Unprocessed Log Queue Growing**  This particular part of the MDI/MPE series is for dealing with ***parsing improvement cases***. KA Parsing Defect (2 of 6) discussed parsing defects and this one follows up with parsing improvements. There are three high-level and general types of parsing improvements: due to **unprocessed log queue,** due to **timeouts**, or parsing **field improvement**. Parts III A, B, and C address them all below.  **PART V of VI: Parsing Improvement For Rule Time Performance**  Part III-C is dedicated to understanding 2052 rule time outs and what to do for them during a support case. We will use case 427933 handled by TSE Dominic Mejia as a model of how to do this type of case and insert additional information conveyed because Dominic was operating from experience where things aren’t always spelled out.  To start this work you will need:  - scmpe.log found in:  -lps\_detail.log found in:  -log samples of timed out logs in llx format  -log samples of timed out logs in csv format  \*Note – The “need” for .csv is relative. It is a nice thing to have and if you can include it please do. If unavailable or if it turns into “for the sake of collection” do not worry about it. LLX can be made to work.  The deep dive covers the history and functionality of timeouts here:  <https://confluence.logrhythm.com/display/GS/DD+-+Data+Processor+-+MPE>  Because the deep dive covers what 2052 timeouts are and their history in the product we will **not** focus this knowledge article on *“what they are”* as much as we will discuss *“what to do about them”.* A one sentence summary of what you can find in the deep dive would look like this: *“2052 timeout errors occur when a particular log has been passed by a specific MPERuleID for more than the default 100ms and as such did not parse correctly”.*  It should also be noted that an email went out from Principal Technical Support Engineer Sean Matthews some time ago regarding Support’s policy regarding the increase of timeout in policy. The entire text of that email is in the additional resources section. The action item to remember is if you find a need to increase the timeout beyond 100ms we should be ultimately asking why. This may include an analysis of customer logs and/or elevating the case to MDI to improve the regex if the use-case can be recreated in a lab.  The above challenges are all addressed in Dominic’s case 427933. You can see that this case was a follow up to an original case 403079. The idea here is that the customer was continuing to see 2052 Timeout Warnings despite MDI informing them that the fix had been published on 25 June 2021 with project P-0012294.  Dominic started this case review by confirming the customer’s concern that there were indeed timeouts in the scmpe.log. There are two ways to do this. You can open the scmpe.log in Notepad ++ and read through the 2052 warning OR you can use the LogAnalyzer tool to gain further insight. You can accomplish the mission with either of these tools but a best practice in my experience is to use them both in conjunction. This example will start with the actual log analysis of the scmpe.log and continue into how the LogAnalyzer tool can be combined to further solve the case. See an example 2052 from Dominic’s case below:  12/29/2021 03:09:20.182634 [CBCLOGRHYTHM-S1] \*\*\*WARNING\*\*\* EVID=2052. MESG=Regex rule match timed out. MsgSourceId = 5372, MPERuleRegexID = 1008299, Base MPERuleID = 1409614, Rule Name = Network Traffic, KB Version = 7.1.628.0, Timeout = 700ms, MsgSourceTypeID = 1000546, PolicyID = -1000546.  While it might be simple reading; it can help to have the anatomy of this log defined. The first line explains itself and we arrive at MsgSourceID= 5372. This should correlate to the LogSourceID on the Log Sources tab to identify which log source had a log that timed out. You can next see the MPERuleRegexID =1008299 as well as the BaseMPERuleID =1409614. These numbers are crucial to know because this is how we identify which rule within the policy is timing out. It won’t be relevant to this case but an additional detail to include regarding the difference between MPERuleRegexID and BaseMPERuleID. At a certain point you will learn to identify when sub-rules are the challenge. MPERuleRegexID can point out a RegexID pattern whereas the BaseMPERuleID will help you in determining if a sub-rule is at play. That said, we digress to the rest of the log.  The other parts of this log are general information which may help troubleshooting. It states the rule’s name in plain text as well as the KB Version. The KB version is critical because MDI releases new KBs every Monday. If a customer is on an older version you might be literally re-inventing the wheel for a problem that has already been fixed. Within the parsing policy itself this is the actual rule that experienced the timeout.  In addition to that information you can find how long the timeout is configured for, MsgSourceTypeID, and PolicyID. The timeout is helpful because it defines whether you are working with a default or tuned rule. The MsgSourceTypeID and PolicyID numbers actually align with Message Sources and Policy IDs inside the EMDB in SQL. It is unlikely you will need that information but it is certainly fun to know if you’re trying to learn how LogRhythm functions on a deeper level.  As discussed above you can also use the LogAnalyzer tool that will give you a perspective of timeouts on a LogRhythm Deployment. A brief explanation of the LogAnalyzer tool will be provided here although there is a separate KA for using this tool. See screenshot 15 and the following explanation to see how it fits into Dominic’s case:  **Screenshot 15**  Table  Description automatically generated with medium confidence  The LogAnalyzer helps by telling you how many 2052 time outs exist and on which rules they exist. On Dominic’s case we can see that the top four MPERuleRegexIDs that are experiencing timeouts were 1010769, 1010505, 1007793, and 1008299. For this case and in this article we will be focusing on MPERuleRegexID 1008299 as it is associated with the Zscaler nano policy. The other rules are not part of this particular case but if you are interested in learning how to read this output in its entirety check out it’s cousin Knowledge Article [Using LogAnalyzer For SCMPE and MDI].  MPERuleRegexID 1008299 is in the 4th position with 13 timeouts. Rule 1008299 is the Network Traffic Rule on the Nano Zscaler policy. Now that we have a specific rule that we are looking at we will want determine how impactful that 13 really is. If you remember back in the scmpe.log’s actual text that the timeout configuration was set to 700? This means that even with 700ms to process there are still 13 timeouts in the log for that time-period. That is bad. In fact, that is horrible. It should be noted that increasing the timeout a little bit can be helpful tuning. If you do this you should note the email from Sean Matthews (former Principal Engineer/Current SRE) regarding MDI elevations. Short version is that if you change the configuration for timeouts beyond 100 then you should be consulting with MPE experts on your team whether the case merits going to MDI. *Changing it and calling it resolved is applying a band aid and not helping LogRhythm maintain a leading edge.*  If you have made it this far then you are most likely evaluating a case for elevation and trying to work out the exact details to send to MDI. This is where you will want to start analyzing the lps\_detail.log. Below you can find where to start with this task.  The first step with the lps\_detail.log is Notepadd ++. Once it is open you will want to scroll all the way to the bottom. Click “search” and check “backwards direction” and type in the appropriate name for the log source. In this case Dominic would have typed in “zscaler”. It is important to note this because the lps\_detail runs in cycles. You will find many entries for zscaler and by doing this you direct your attention to the most recent. You could also simply use the log called lps\_detail snapshot and avoid that entire process.  Once you have done the above you should be looking at something that resembles screenshot 16.  **Screenshot 16**  Table  Description automatically generated  Screenshots 16 shows us a lot of information. The first thing we want to look at per the deep dive in an LPS\_Detail is the Total Compares and the LPS-Policy-Total. Read the deep dive for depth but a great line in there was : *A general rule of thumb is to look for a LPS-Policy-Total that is below 100 for logs that are greater than 10,000 compares.* **Another way of looking at that using math is to answer the following question: is LPS\_Policy Total less than 1% of the Total Compares?**  Observe in screenshot 16 that 1% of 971,311 is 9,713. Our LPS-Policy-Total is currently at 1,962. This math should be considered as the deep dive suggests “a rule of thumb”. It is not completely accurate. If you are getting numbers here in the 6,7, or 8,000 range then it’d probably be a volume thing. In this case though we can say that 1,962 is for sure quite low.  We can also see that autosorting is not a problem with this Zscaler policy because the “A” in the third column has moved the MPERegexID 1008299 to the very top. This in turn though has created a bottleneck in some ways. Bottlenecks can appear anywhere in the policy and our role in support is to identify where they exist; MDI will fix them. In this case we can identify the difference between “bottle neck” and “properly sorted autosort rule being at the top” because it is sorted at the top and the ratio of total compares and LPS\_total is not within that 100:10,000 provided by engineering.  ***This is also supported by the scmpe.log which was showing timeouts with the policy set to 700ms time out.***  The next steps for this particular case are set: an elevation to MDI was needed and that is what Dominic pursued. It might also be helpful to harvest some more fun facts out of screenshot 16. You can see that after the initial 971,311 logs parsed that there were 6 logs left that did not match that first rule. These then fall to the next rule and the next until they match. You can see that some matched on the catch all level 4 and the rest on the catch all level 1. If logs make it to the bottom and have not been sorted that is where we get unclassified logs and a return to the other types of parsing cases in this KA series.  **NEXT STEPS AND ACTION ITEMS WHEN YOU ARE IN THIS POSITION**  **Parsing Improvement**    **Existing customer or proof of concept? Existing Customer**    **Current Log Processing Policy Version: (Required)**  **LogRhythm Default**    **Use case or justification for the request (Required)**  **Customer is seeing consistent timeouts on Syslog – Zscaler Nano Streaming Service. Time Out is currently configured to 700 MS and timeouts continue. In addition LPS-Policy-Total and Compares are about ¼ of what the should be according to the deep dive.**    **MPERuleRegexID to be improved? 1008299**    **The customer MUST be on the latest version before we can elevate to MDI.**  **7.1.621**  **Does the logging format adhere to our standards (e.g., the correct IIS fields are enabled, etc.)?**    **Yes**  **Actual parsing behavior: Parsing is slow enough to create timeouts.**    **Desired parsing behavior: Faster parsing to improve timeouts.**     * Log samples are required - Included   HAVE YOU ENTERED THE LOGSOURCE TYPE INTO THE SUPPORT CASE? Yes  **You would then follow the SOP set at this link which would have you move it into the appropriate MDI queue.**  <https://confluence.logrhythm.com/display/GS/MDI+Case+Handling#MDICaseHandling-ParsingImprovementRequest>  -------  Side Note After Case Was Elevated  This would be out of the realm of support but as a piece of interest as to the question: what happens when it leaves my queue?  You can follow the case progress and you can see that MDI first tackled this with project (P-0012679) and then again with P-0012731. The second of those projects seemed to be a permanent fix according to the customer. |
| Remediation Process |
| Root Cause |
| Additonal Resources  **Email From Sean Matthews on topic of MPE Time >100**  *Sending this wide and far as this setting is being increased globally.*  *The MPERuleTimeout value is set to 100MS per log processing policy by default, this is a very generous value. You should not be increasing this unless you are extremely sure it needs to be increased in order to parse critical logs.*  *Typically a good performing MPE rule will take <40MS to parse logs against it, normally less than this. So 100MS is a very generous amount of time.*  *Increasing this timeout value is going to have a severe impact on the deployment. Do Not go and increase it just because you see timeouts in the scmpe.log, timeouts are good, that is what the setting is there for, to speed up the processing speed. By increasing this timeout you are basically giving it longer to timeout each individual MPE rule which in turn makes performance of the DP much worse.*  *Should you feel the need to increase the MPERuleTimeout setting for a Log Processing Policy, you should be raising a case with MDI to improve the performance of the MPE rules, with the intention to turn it back to the default 100MS.*  *I am currently seeing many deployments with this set to 400MS or even up to 1000MS for log processing policies, this is criminal, you risk slowing down processing performance on the Data Processors, which in turn creates an unprocessed logs queue, which in turn means a delay in processing and potential risk of logs not being observed against AIE use cases.*  *I cannot stress this enough, do not increase this value unless you are 100% sure you absolutely need to and you have a plan to bring it back down to 100MS.*  A really great introduction to how regex works…not mandatory but will for sure help you understand these concepts.  <https://coralogix.com/blog/regex-101/>  A great tool to test regex on logs that is fairly user-friendly.  <https://regex101.com/>  A guide to the top 20 Regex Strings  <https://regexland.com/most-common-regular-expressions/> |
| Tags |