|  |
| --- |
| Title : Parsing Defects/Parsing Improvements: Where To Start When You Don’t Know |
| URL Name |
| **PART III 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 III of VI A: Unprocessed Log Queue Growing**  The first of the parsing improvement topics is when the unprocessed log queue is growing. When unprocessed log queue is growing are two common causes for this: undersubscription or poor parsing. Logs that end up in the **STATE** directory at ***S:\Program Files\LogRhythm\LogRhythm Mediator Server\state\UnprocessedLogs*** can be there because the log volume as seen in performance monitor is too high OR it could be growing because a poorly performing parsing rule is slowing down the processing.  A great example of how to identify when parsing is the challenge is case 432574 by Grant Hogan. You can see that on a call he confirmed logs were spooling to that directory and then he went back to get a historical view of when unprocessed logs started to climb. Take a look at screenshot 5 from his case.  **Screenshot 5**  Chart  Description automatically generated  It should be pretty simple to see that starting on 15 December there was a massive jump in unprocessed logs. Less clear, but very much in the picture is the height of the bars themselves. It’s always important to take log volume into consideration. In this example log volume is relatively unchanged so we can rule that out as part of the analysis/solution. You can also evaluate log volume using performance monitor. The next step will be to figure out which of these logs are exactly “unidentified”. See screenshot 6 below  **Screenshot 6**  **Text, timeline  Description automatically generated**  **The above screenshot is how you would configure an investigation to find unidentified logs.** One thing to keep in mind about processing and mediator flows is that what you are seeing are logs that have been indexed. That is to say they have been indexed without a Classification/Common Event. This occurs after the 2052 time out error is created to alert us of challenges with processing. If you have spooled logs in the state directory listed above it will be important to note that those ARE NOT represented in this data. Those have yet to error out and parse as blank classifications or common events. You can see exactly how they parse in screenshot 7 which is from Grant’s case. He would have gotten this by doing the above investigation, going to log viewer, and double clicking on a log.  **Screenshot 7**  Graphical user interface, text, application, Word  Description automatically generated  Notice how there is a full log message, but classification and common event are missing. The next step for this type of scenario is to replicate it on your VM. **Before you get off the call** with the customer make sure that you have done both “export all logs” and “send all logs”. One will create a CSV and one will create an LLX. You may want both of them in your troubleshooting. If you haven’t gathered the scmpe.log or LPS\_Detail.log then you will want to do that as well.  You will want to open client console in SuperUser mode. Open MPE Rule-Builder and use the yellow folder to select the proper policy. In this case you should be selecting Syslog-Fortinet Fortigate v6.0. The important part here will be to select the “test all” button as show in screenshot 8. Don’t worry, the screen will click but otherwise will look as if you made no changes. This is what happens while test a whole policy.  **Screenshot 8**  Graphical user interface, application, table  Description automatically generated  Once you have loaded the policy you will want to load in your llx. See screenshots 9 + 10 below. If you did it correctly it should look like that.  **Screenshot 9**  Graphical user interface, text, application  Description automatically generated  **Screenshot 10**  Graphical user interface, application, table  Description automatically generated  Next step is to actually test it. Click the button that says, “test all”. You will notice MPE Rule Builder processing for a minute (or several) to which it will produce a screenshot like the one from Grant’s case.    **Screenshot 11**  Graphical user interface, application, table, Excel  Description automatically generated  The most important thing to observe in the results above is how the llx sample did not hit on any of the parsing rules except for the catch-all level 3. You can see this in the red box in screenshot 11. We would have expected these logs to hit on the second to last rule called “UTM : SSL Message”. How we know this is the rule will be described later. For now, the important part to digest is that the regex of that rule clearly did not catch them and connecting to the lesson of catch-alls above we can understand that this log will be identified but in need of improvement.  Once you have identifed that the logs are all falling to the catch-all your next step will be to determine which rule they should have belong to. You can identify logs by studying them and asking the customer. An example of the log in Grant’s case looks like this:  12 16 2021 06:02:55 10.21.7.6 <LOC7:NOTE> date=2021-12-16 time=06:02:54 devname="s5652a" devid="FGT6HD3916804855" eventtime=1639634575050932732 tz="+0000" logid="1700062302" type="utm" subtype="ssl" eventtype="ssl-anomalies" level="notice" vd="s5654a" action="resign-as-untrusted" policyid=1034 sessionid=1062819319 service="HTTPS" user="U115969" group="Domain-Users" profile="SSL\_Inspection\_HTTPS" srcip=172.27.69.211 srcport=58744 dstip=20.54.89.106 dstport=443 srcintf="port12" srcintfrole="undefined" dstintf="port13" dstintfrole="undefined" proto=6 eventsubtype="certificate-anomaly" msg="Server certificate is re-signed as untrusted, certificate-status: untrusted." hostname="slscr.update.microsoft.com"  The log above has a key part highlighted in yellow. There are only so many rules where that can be match. SSL messages would be an appropriate hypothesis. Often you can simply ask the customer and system admins will know their logs better than anyone. They’ll tell you what it is.  You can double check the parsing of the llx against the singular rule directly. This is a great thing to check because it can help determine if there is something wrong with the policy or the rule. This is all information that is helpful to the MDI team. You do this by loading the individual rule in and click test. You’ll get something like screenshot 12 below:  **Screenshot 12**  Graphical user interface, text, application  Description automatically generated  Notice how in screenshot 12 that none of the fields parse. This is due to the 99% non-match rate. **Your role here is to identify where they should be parsed and prove that they’re not being parsed that way, thus an improvement.** MDI will do the actual changes to the regex. This screenshot also gives MDI an idea of timing for this log processing. With this information your next step will be to complete the template for MDI-Elevation. Grant’s completed template can be found below. Template is in **purple** while Grant’s responses are in **green**.  **Parsing Improvement**    **Existing customer or proof of concept?** Customer    **Current Log Processing Policy Version: (Required)**  LogRhythm Default  **Use case or justification for the request (Required)**   The customer has been seeing numerous regex timeouts for the Syslog – Fortinet Fortigate v6 log source.  Upon investigating this we could see that there was a number of unidentified logs coming into the deployment.  The issue started on 16/12 which was just a day after the 7.1.628 KB was released so it appears the KB update has changed how this was parsed.  When replicating this issue in my lab I could see I was on KB version 7.1.625 so I tested it with version and could see the logs were being matched on a MPE rule called ‘Catch All : Level 3’.   I then updated my KB to 7.1.628 to match the customers deployment and could see that the logs didn’t match on any of the MPE rules. I could also see that the ‘Catch All : Level 3’ from KB version 7.1.625 was no longer present.  **MPERuleRegexID to be improved?**   The logs are not being hit by any MPE rule and I no longer have the regex ID from the catch all rule from the 7.1.625 KB.  **The customer MUST be on the latest version before we can elevate to MDI.**  **Ensure the KB Version field is populated.** 7.1.628    **Does the logging format adhere to our standards (e.g., the correct IIS fields are enabled, etc.)?** Yes    **Actual parsing behavior:** Screenshots shown above.    **Desired parsing behavior:** For logs to be parsed and not come through as unidentified.     * Log samples are required * Ensure log samples only contain logs that need the parsing improvement (in investigator, filter for non-matched logs > right click > send selected logs)   Log samples are attached in the case    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> |
| Remediation Process |
| Root Cause |
| Additonal Resources  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 |