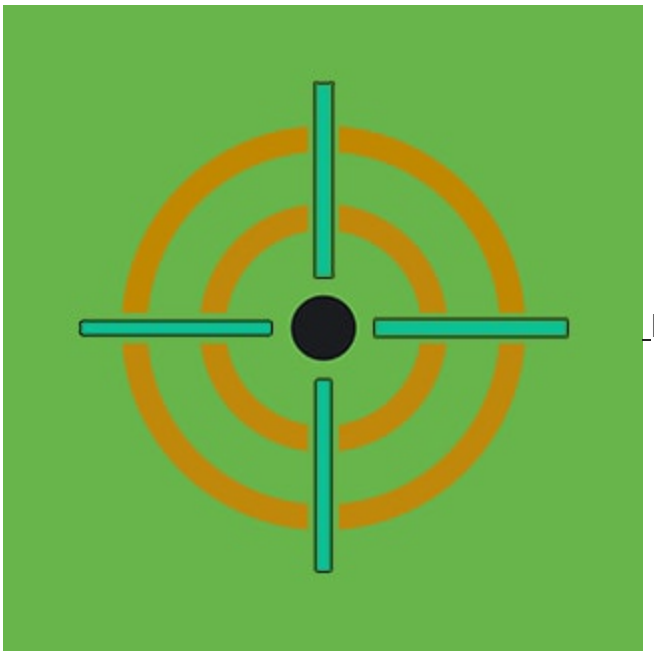
ML Detection of Risky Command Exploit

splunk.com/en_us/blog/security/ml-detection-of-risky-command-exploit.html

July 26, 2022



By Splunk Threat Research Team July 26, 2022

As described in Splunk Vulnerability Disclosure SVD-2022-0624, there is a list of SPL (Search Processing Language) commands that are classified as risky. This is because incorrect use of these risky commands may lead to a security breach or data loss.

As a precautionary measure, the Splunk Search app pops up a dialog, alerting users before executing these commands whenever these commands are called. However, there are scenarios where this safeguard measure can be bypassed, leaving a vulnerability to malicious users to exploit these risky commands to gain higher privilege, to collect security data or to delete data queried.

Although rules can be defined to find these risky command searches, it is difficult, if not impossible, to identify the searches that maliciously exploit these vulnerabilities without incurring large amounts of false positives. It is therefore desirable to develop methods to detect such risky command misuse or abuse using machine learning (ML) algorithms — in addition to rule intelligence detections — to further pinpoint a true threat.

One of the targets of malicious exploits of these risky commands is to exfiltrate data. Expecting an unusually long run time compared with benign searches containing risky commands, we therefore can assume that the search time anomaly is an indicator of the exploit of risky command vulnerability from attackers. Based on this assumption, we developed a machine learning approach to model users' behavior of search run time with risky commands and detect such anomalies to alert analysts of possible threats.

This is accomplished by using the time spent executing one of these risky commands as a proxy for misuse/abuse of interest during an investigation and/or hunting. The detection builds a model utilizing the MLTK DensityFunction algorithm on Splunk app audit log data. The model is trained from users' historical reference of running the risky commands, and then the total search run time of executing these commands in each hour is aggregated as an indicator of user behavior to perform anomaly detection.

Implementation

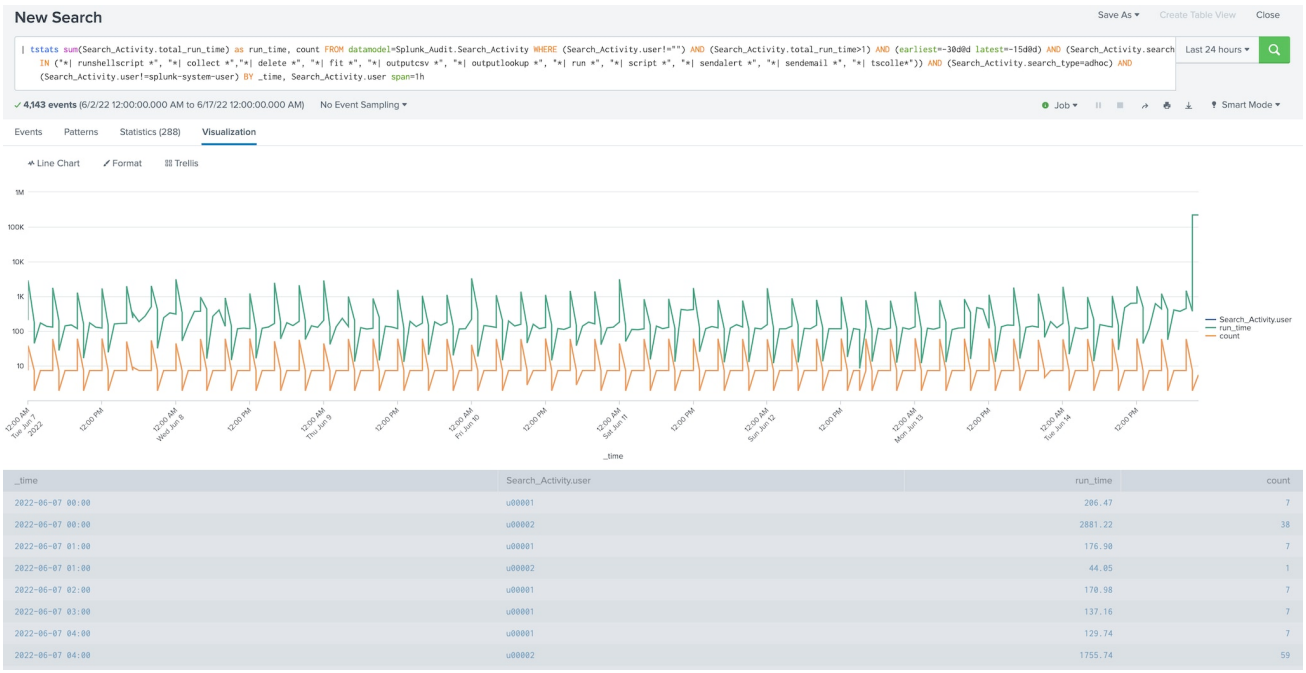
We build our detection based on Splunk app audit data, specifically search activities in the audit data model. The related data fields used in this detection are:

- search: the search string
- search_type: the type of the search
- user: the name of the user who ran the search
- total_run_time: the time used to run the search

Where 'search', 'search_type' and 'user' fields are used as filters to narrow the detection scope to be correlated with risky command vulnerability exploits, and the 'total_run_time'

field is used to model user behavior. We process the log by ‘bin’ command to aggregate them into hourly intervals to suppress noise. This operation can generate another numerical field ‘count: the number of runs’.

Exploring the data closely, we notice that the trend of these two numerical variables is heavily correlated to each other as shown in the below figure because both are derived from the same log events, we thus choose ‘total_run_time’ as a single indicator. In this way, we can use MLTK DensityFunction as our underlined algorithm since at the current time this algorithm works only for univariate data.

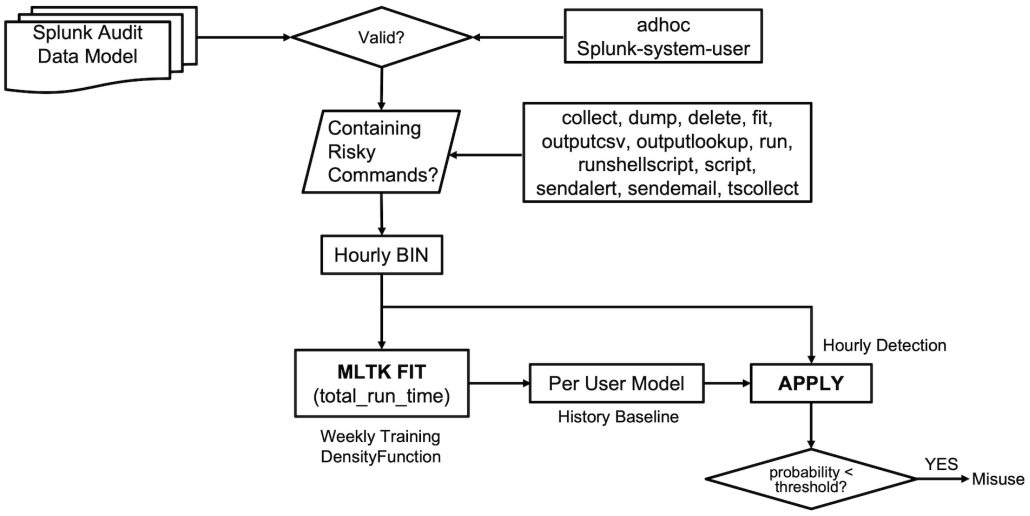


_time	Search_Activity.user	run_time	count
2022-06-07 00:00	u00001	206.47	7
2022-06-07 00:00	u00002	2881.22	38
2022-06-07 01:00	u00001	176.90	7
2022-06-07 01:00	u00002	44.05	1
2022-06-07 02:00	u00001	170.98	7
2022-06-07 03:00	u00001	137.16	7
2022-06-07 04:00	u00001	129.74	7
2022-06-07 04:00	u00002	1755.74	59

In our detection, the ‘total_run_time’ of past 7-day data is fed to the MLTK app ‘FIT’ command to train a baseline model of user behavior, along with ‘user’ as a ‘by’ clause to create per-user models. The model can then be used to monitor new search activities continuously.

By using ‘APPLY’ command to infer whether ‘total_run_time’ in the last hour of a user is an anomaly, the model will alert a potential exploit of risky commands. The overall detection flow is presented in the below diagram. The model identifies the top 0.1% of user search run time, which signals a potential exploit of these risky commands. Users can adjust this threshold to values higher or lower than 0.1% as needed to adjust the efficacy of the model based on the acceptable true positive/false positive rates.

Users can also choose to modify baseline build intervals, currently set at 7 days, depending on the search activity frequency in their environment. The principle is that the data points used to build baseline should be large enough so that the baseline model represents users’ normal behavior.



As shown in the above flow chart, our detection:

1. uses unsupervised machine learning algorithms, therefore no labeled data is required to train the baseline models
2. builds a per-user baseline, therefore misuse is determined based on each user's behavior history data

Run time of search activities varies dramatically among users as shown in the below data exploration sample where the standard deviation is as large as 779. The data distribution, which determines normal behavior and impacts model performance, is unable to be predefined before actual user data is collected, we therefore set the 'dist' parameter of DensityFunction in our detection baseline training as 'auto' so that the algorithm can learn the best distributions from each user's behavior, though it will take much longer model training time because the process will have to train a model for each distribution and select the one with the best performance.

Users can modify earliest=-7d@d in the search to other values so that the search can collect enough data points to build a good baseline model. Users can also modify a list of risky commands in "Search_Activity.search IN" to better suit users' violation policy and their usage environment.

users: 28

Top 10 Values	Count	%
splunk-system-user	2,343,735	82.069%
admin	331,644	11.613%
internal_observability	173,301	6.068%
de [Full Name Masked]	3,835	0.134%
da	705	0.025%
hf	595	0.021%
dd	331	0.012%
xl	308	0.011%
ny	239	0.008%
mm	178	0.006%

total_run_time

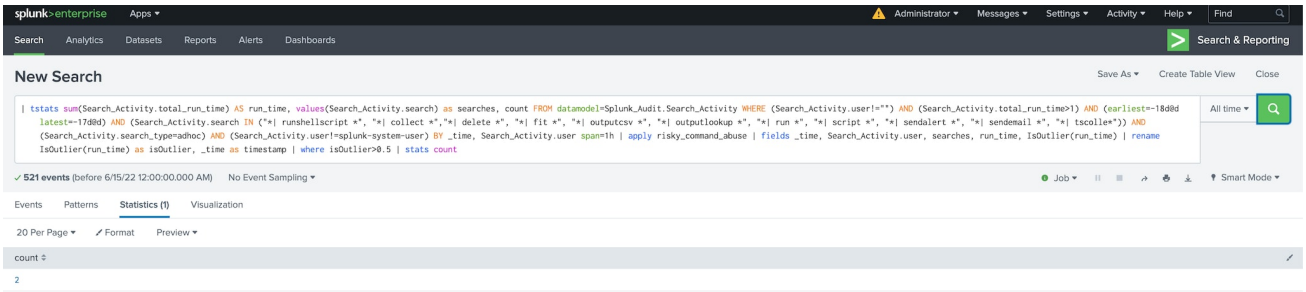
Avg: 30.543817727139544 **Min:** 0.00 **Max:** 473414.89 **Std Dev:** 779.2151088427181

Top 10 Values	Count	%
0.01	169,176	10.48%
0.04	87,611	5.427%
0.06	49,346	3.057%
0.00	32,118	1.99%
0.05	30,652	1.899%
0.02	28,563	1.769%
0.03	22,966	1.423%
0.10	16,832	1.043%
0.11	15,340	0.95%
0.21	13,429	0.832%

Also, we set ‘lower_threshold’ to a tiny value (0.000001) so that the lower bound of the anomaly is sufficiently close to zero and no search activity with short runtime will be wrongly marked as positive.

```
| fit DensityFunction "run_time" dist=auto lower_threshold=0.000001
upper_threshold=0.001
```

To test our implemented detection, we manually implanted two anomalies into a synthesized dataset of two users (one anomaly for each user). As shown in the below figure, these two anomalies are reported in the inference stage and matched our expectation.



Applications

The corresponding detection in ESCU is "Splunk Command and Scripting Interpreter Risky SPL MLTK". To use this detection, Splunk accelerated audit data model must be available. Detection should be scheduled to run hourly to detect whether a user has run searches containing risky SPL with abnormally long running time in the past one hour, compared with his/her past seven days history. This detection depends on the MLTK App that should be installed before running this detection. The list of apps this detection depends on:

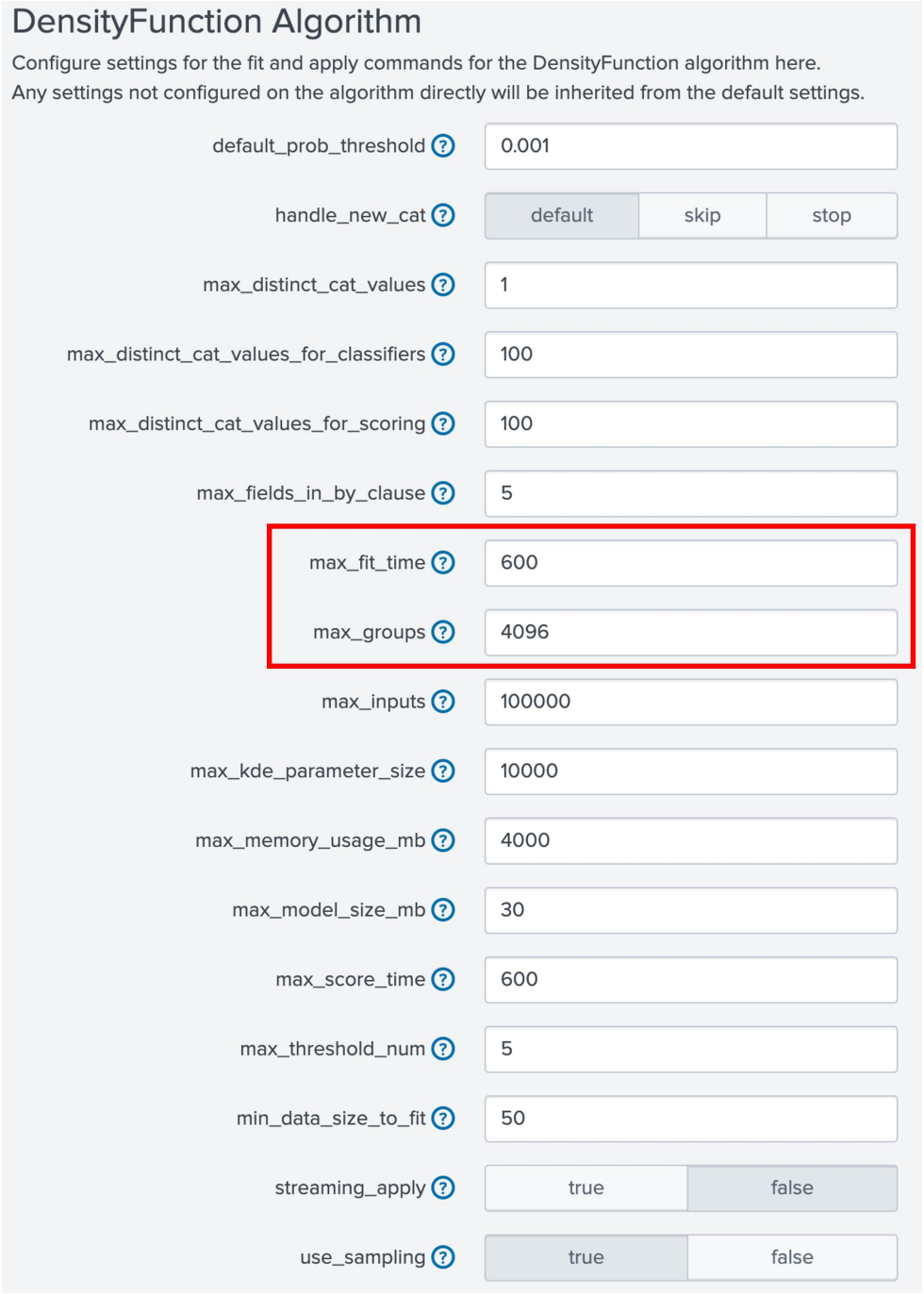
- Splunk Machine Learning Toolkit
- Splunk Common Information Model

- Python for Scientific Computing

The name of the machine learning model generated by this detection's baseline training is "risky_command_abuse" and should be configured to be globally shared (not private) in the MLTK app as described in the MLTK document unless the same account of training this model will be used to perform inference using this model for anomaly detection.

For large enterprises, for example where more than 1,000 users will actively run Splunk searches, training the baseline model might take significant computing resources and might require a dedicated search head. Default settings of this detection's underlying DensityFunction algorithm within MLTK App may need to increase to achieve optimal performance as described in the section Configuring DensityFunction parameters section in the manual for MLTK App, especially for these parameters:

- max_fit_time: maximum time allowed to run FIT command to train baseline models
- max_groups: maximum number of users who run searches with risky commands
- min_data_size_to_fit: minimum number of data points to train a baseline



Learn More

If you would like to adopt this detection, you can get the corresponding baseline and detection YAML file from the Splunk Security Content GitHub repository.

Type	Name	Technique ID	Tactic	Description

Baseline	Splunk Command and Scripting Interpreter Risky SPL MLTK Baseline	T1059	Execution	This YML is to build baseline models for risky command exploit detection from user's past 7 days' search activities using total search run time as user behavior indicator.
Anomaly	Splunk Command and Scripting Interpreter Risky SPL MLTK	This YML is to utilize the baseline models and infer whether the search in the last hour is possibly an exploit of risky commands.		

More Related Detections

Hunting	Splunk Command and Scripting Interpreter Risky Commands	T1059	Execution	This YML file is to hunt for ad-hoc searches containing risky commands from non-administrative users.
Anomaly	Splunk Comma and Scripting Interpreter Delete Usage	This YML is to identify the use of the risky command 'DELETE' that may be utilized in Splunk to delete some or all data being queried.		
Anomaly	Detect Risky SPL using Pretrained ML Model	This YML is to use a pre-trained machine learning text classifier to detect potentially risky commands.		

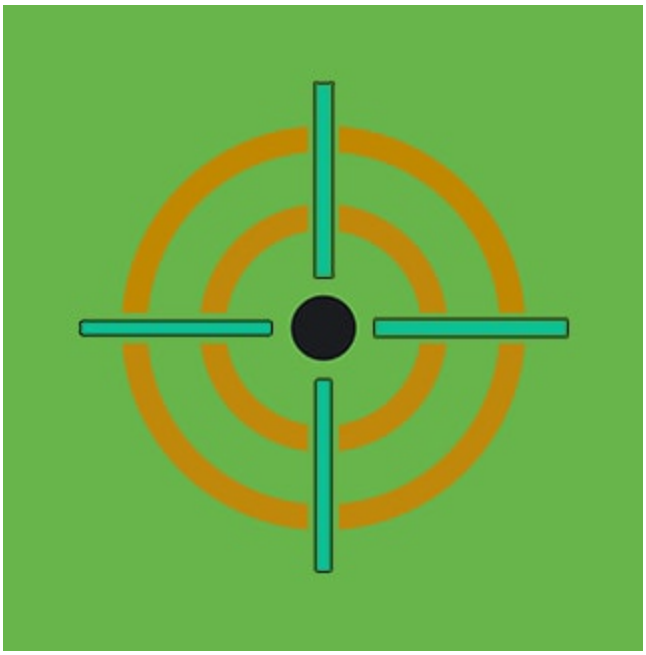
Feedback

Any feedback or requests? Feel free to put in an issue on GitHub and we'll follow up. Alternatively, join us on the Slack channel #security-research. Follow these instructions If you need an invitation to our Splunk user groups on Slack.

Acknowledgments

We would like to thank the following for their contribution to this post and corresponding detections:

- Abhinav Mishra
- Eric McGinnis
- Glory Avina
- Jose Hernandez
- Kumar Sharad
- Michael Haag
- Rod Soto
- Xiao Lin



Posted by

Splunk Threat Research Team

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by providing tactical guidance and insights to detect, investigate and respond against the latest threats. The Splunk Threat Research Team

focuses on understanding how threats, actors, and vulnerabilities work, and the team replicates attacks which are stored as datasets in the Attack Data repository.

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more Splunk Security Content.