Collaborative Information Sharing for ML-Based Threat Detection

Talha Ongun* Simona Boboila* Alina Oprea* Tina Eliassi-Rad*
Alastair Nottingham† Jason Hiser† Jack Davidson†

Abstract

Recently, coordinated attack campaigns started to become more widespread on the Internet. In May 2017, WannaCry infected more than 300,000 machines in 150 countries in a few days and had a large impact on critical infrastructure. Existing threat sharing platforms cannot easily adapt to emerging attack patterns. At the same time, enterprises started to adopt machine learning-based threat detection tools in their local networks. In this paper, we pose the question: *What information can defenders share across multiple networks to help machine learning-based threat detection adapt to new coordinated attacks?* We propose three information sharing methods across two networks, and show how the shared information can be used in a machine-learning network-traffic model to significantly improve its ability of detecting evasive self-propagating malware.

1 Introduction

With the increased connectivity of devices on the Internet, attackers have an opportunity to launch coordinated global attacks targeting multiple networks. Self-propagating malware (SPM) such as Mirai [5] and WannaCry [18] infected hundred of thousands of machines and caused significant damage on a global scale. More recently, a widespread campaign in the supply chain of the SolarWinds remote monitoring software infiltrated thousands of organizations around the world [4].

Public threat intelligence platforms such as Malware Information Sharing Platform (MISP) [1] provide a database of malware signatures and indicators of compromise (IoCs) to enable network owners to deploy solutions to protect against known threats. These platforms can be used to detect known threats, but they fail to adapt quickly to new attack campaigns. Moreover, adversaries can easily create many malware variants in an attempt to evade static signature detection. Recently, machine learning-based threat detection techniques have been proposed (e.g., [23, 7, 21, 15]) and they have started to be widely adopted in the industry [3, 2]. Most of these ML systems train models locally on the enterprise network, and attempt to detect attacks using the security logs of a single enterprise.

In this paper, we pose the question: *What information can defenders share across networks to help machine learning-based threat detection adapt to new coordinated attacks?* Our hypothesis is that ML-based tools can significantly increase their detection ability by leveraging attack information shared across multiple defenders. We demonstrate the advantages of information sharing via a case study of detecting SPM attacks using the PortFiler network traffic system [14]. We propose and analyze three methods for information sharing, including (1) sharing an entire ML model; (2) sharing weights for an ensemble model; (3) sharing weights for an ensemble model and feature distribution information. All of these methods share aggregated information and protect the privacy of security logs, an important consideration in sharing threat information. Our analysis shows that a locally-trained ML ensemble can detect more evasive malware with 0.91 precision at 0.86 recall on average using shared threat information and feature distribution information. We conclude by discussing the challenges on designing and deploying global defensive ML models to counteract coordinated attacks.

2 Methodology

We provide an overview of the PortFiler system designed to proactively detect SPM attacks in an enterprise network [14]. We discuss our machine learning (ML) methodology, and three information sharing approaches to improve local detection. The overview of the information sharing platform is given in Figure 1.

2.1 PORTFILER System Overview. PortFiler [14] is an unsupervised ML threat detection system that uses Zeek network connection logs [17] collected at the border of the enterprise network. This data contains connection metadata information such as timestamp, duration, source and destination IP and port, transport protocol, payload size, number of packets, and connection state. PortFiler monitors a set of configurable ports, including port 23 (Telnet)

*Northeastern University, Boston, MA, USA
†University of Virginia, Charlottesville, VA, USA

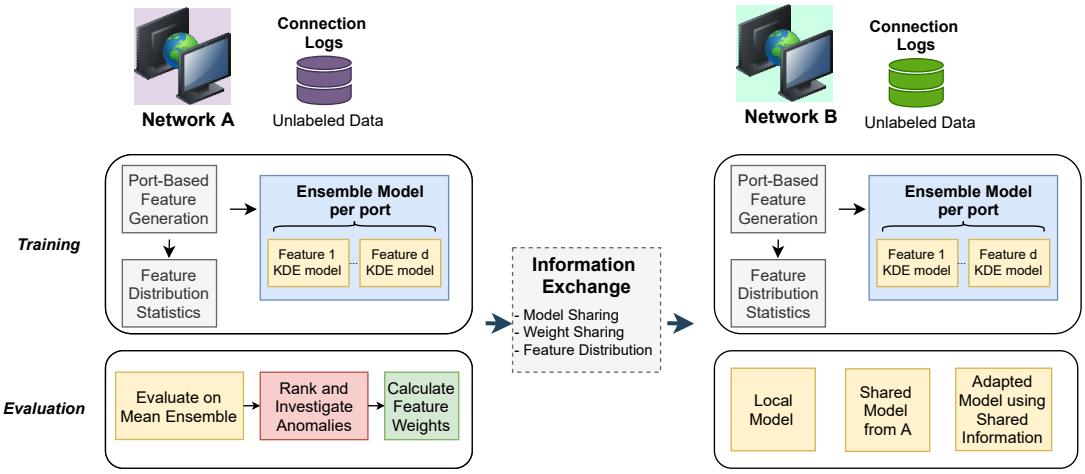


Figure 1: Overview of the information sharing platform where NET-A shares either its local ML model, feature weights for the ensemble, or feature distribution to improve the local ML model of NET-B.

used by Mirai, port 445 (SMB) used by WannaCry, as well as ports 22 (SSH), 80 (HTTP), and 443 (HTTPS).

PORTFILER learns the distribution of network traffic on each port during the training period, using a set of 35 statistical traffic features aggregated by one-minute time windows. Ensembles of KDE models are shown to be the most resilient to evasion [14]. The models that are part of the ensembles are trained on individual features. During testing, each model generates an anomaly score. All scores are combined into a final score using: (1) uniform weights (*Mean Ensemble*); or (2) different weights (*Weighted Ensemble*).

2.2 Information Sharing for SPM. SPM attacks rarely target a single network or organization, due to their ability to quickly propagate across the Internet. Our main hypothesis is that organizations can share threat information after SPM attack detection to help other organizations improve their detection. We propose several scenarios of sharing information between networks. Privacy is a major consideration for threat information sharing and all our methods protect the privacy of security logs, as we only share aggregated information on network traffic, as described below:

Model Sharing. NET-A trains the ML model on its network logs, and shares the model directly with NET-B. This approach saves training time at NET-B, but differences in traffic patterns between the two sites may render the transferred model less efficient in capturing anomalies at NET-B.

Weight Sharing. NET-A and NET-B train two ensemble models independently on their own logs. Assuming that NET-A is detecting an SPM attack, it computes feature weights based on the detected samples and shares them with NET-B.

To compute feature weights, NET-A applies a Random Forest classifier trained on the malicious samples, as well as legitimate samples to derive feature importance. The feature weights will then be normalized values of feature importance, summing up to 1. NET-B uses the Weighted Ensemble trained on its own network traffic, with the model weights shared by NET-A. Intuitively, the features that are most relevant for the detected attack get assigned higher weights, and will contribute more in the anomaly score generated by the Weighted Ensemble at NET-B. We compare the Weighted Ensemble with shared weights to the unsupervised Mean Ensemble with equal weights at NET-B.

Weight Adaptation. Feature distributions on the two networks can be significantly different, and, thus, some feature weights do not transfer effectively. To alleviate this effect, we propose to share information about feature distribution from NET-A, in particular the first four moments of distribution of each feature values: mean, variance, skewness and kurtosis. NET-B compares the distribution of its own features to those of NET-A and selects only the closest features and their weights in the Weighted Ensemble. The method is outlined below:

1. Each network computes its moments of distribution: $M_A[X]$, $M_B[X]$ for each feature vector X.
2. NET-A computes the feature weights.
3. NET-A shares its feature weights and moments of distribution with NET-B.
4. NET-B computes Euclidean distance between the moments of the two networks: $||M_A[X] - M_B[X]||_2$

Finally, NET-B selects top k (i.e., $k = 10$) features ordered by distance and carries out detection using only these top features.

3 Evaluation

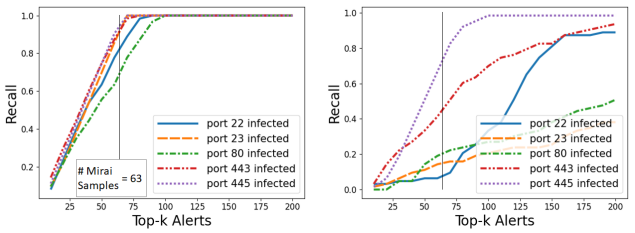
3.1 Dataset We used Zeek connection logs collected by University of Virginia (UVA) and Virginia Tech (VT). Our experimental setup uses one week of training and one day of testing at each network on July 2020, 9.64 billion events for UVA, and 9.69 billion events for VT. The dataset was anonymized to not reveal personal information about the machines or users on the network[1]. To evaluate our system, we merge malicious Mirai traces [16] at testing time on different ports. We generated an evasive variant of Mirai, 128 times slower than the original, by sampling a fraction of connections to reduce the scanning rate. Ongun et al. [14] showed that scanning speed may differ among prevalent SPM families.

3.2 Experiments We simulated a scenario where NET-A (UVA) is infected by the original Mirai malware. NET-A runs the unsupervised Mean Ensemble model of PORTFILER and detects this attack with accuracy above 0.96. We assume that NET-B (VT) is infected later either by a similar fast Mirai variant, or by the 128x slower variant. We evaluated the three proposed information sharing methods from NET-A to NET-B, where PORTFILER is also deployed with the same set of features. We start our evaluation with a baseline model where no information sharing is employed and then compare the three sharing methods.

Baseline. We consider the Mean Ensemble of KDE models trained on NET-B, without any shared information. Figure 2 shows the recall in the top k alerts for the fast and slow Mirai variants. Our test data contains 63 malicious samples (one-minute time intervals) out of 1440 in total. Ideal recall in top k is k/m, where m is the number of malicious samples. Therefore, an ideal model would rank all 63 malicious samples on top, without any false positives. This translates into a recall of 1.0 after processing the top 63 alerts. We represent the number of Mirai samples in our plots with a vertical line. The baseline model performs well on the fast Mirai variant (Figure 2a). However, its performance is much reduced on the slow Mirai variant (Figure 2b) on most ports.

Model Sharing. Here, NET-A shares the entire ML model (Mean Ensemble) and the results at NET-B are shown in Figure 3. The fast Mirai variant is detected generally well, except on port 445. This is due to different background traffic patterns at the two sites. NET-A blocks most of the traffic on port 445, while

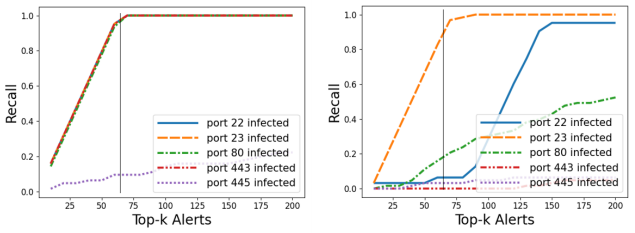
[1]The IRB office reviewed our data collection process and determined that our research does not qualify as Human Subject Research.



(a) Fast Mirai variant (b) Slow Mirai variant

Figure 2: Baseline at NET-B, the accuracy is low for slow variant with no shared information. The vertical line represents the number of Mirai samples.

NET-B still allows internal traffic on this port. This leads to an important observation: on some ports, the model is not directly transferable between sites. Prior analysis of the traffic patterns is necessary to establish whether the model can be shared. For the slower Mirai variant, we generally see a performance degradation when the model is shared compared to the baseline. At slower malware propagation speeds, the detector is more sensitive to traffic variations between the networks.

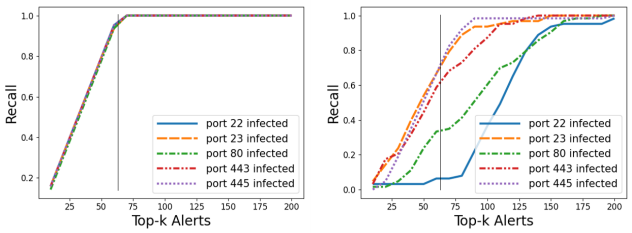


(a) Fast Mirai variant (b) Slow Mirai variant

Figure 3: Model Sharing. Traffic pattern variations between the two sites where training and testing are done decrease accuracy. The vertical line represents the number of Mirai samples.

Weight Sharing. Each site trains an ensemble model on their own data. NET-A uses the Mean Ensemble detector to rank and label malicious time windows and derive feature weights. These weights are shared with NET-B, which uses its own trained model, along with these shared weights to detect anomalies in the test data. Figure 4 shows the detector's performance on NET-B test data. This method is able to correctly identify malicious time windows better than either the baseline or the model sharing scenario. It obtains almost perfect detection on the fast Mirai variant, and generally better results on the slower Mirai variant. However, on ports 22 and 80 the results can be further improved.

Weight Adaptation. We take the weight sharing approach one step further by adjusting the weights based on the statistical distance between feature distributions



(a) Fast Mirai variant (b) Slow Mirai variant

Figure 4: Weight Sharing. Detection is improved by giving higher weights to features that are more relevant to a particular attack.

at the two sites and using only the 10 closest features. The fast Mirai attack is detected with maximum recall and precision and thus the graph has been omitted. Figure 5 shows the system's ability to detect malicious time windows at NET-B on the slow Mirai attack. The recall metric is improved across all ports, illustrating the benefit of an adaptive approach for weight sharing. Precision and false positive comparison of sharing methods are presented in Table 1 and 2 of Appendix B.

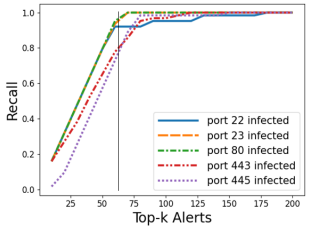


Figure 5: Weight Adaptation (slow variant). This method performs best, by adapting the shared weights based on the statistical distances between features at the two sites.

4 Related Work

Vasilomanolakis et al. [20] provide a taxonomy of collaborative intrusion detection systems. Earlier work on distributed intrusion detection leverages matching attack graph signatures to detect coordinated attacks [22, 13]. Alert correlation has been proposed to reduce the number of false positives by sharing alerts centrally to confirm malicious behavior [8, 6]. Hu et al. [10] uses boosting classifiers for distributed threat detection. Parameterized local models are collected to build a global model improving the detection rates. Nguyen et al. [12] study the first federated-learning-based anomaly detection system for IoT networks.

Threat intelligence sharing has been studied to consider different challenges such as efficient coordination, addressing legal regulations, and standardization efforts [19, 11, 9].

5 Discussion and Conclusion

We proposed three information sharing methods across two networks and showed how the shared information can be used in an ML model to significantly improve its ability to detect evasive self-propagating malware. We showed that network traffic distribution differs significantly per port across organizations and direct model transfer is not effective. While weight sharing works better than model sharing, the weight adaptation method, which selects feature weights according to the closest features in the two networks, performs best at detecting the evasive variant.

We highlight the need for global defensive models that can benefit from information sharing across defenders to better protect against coordinated adversaries. We discuss several challenges and open problems to realize this vision:

Generalization and Evasion. Although we show the effectiveness of our approach against SPM attacks, generalization to other attacks remains an open question. Can the ensemble model be used with different weights to adapt to other attacks? We have shown resilience to evasion against an adversary slowing down the propagation rate, but how resilient are these methods against more advanced evasive strategies?

Threat intelligence sharing. Platforms such as MISP [1] facilitate threat sharing of IPs, domains, and malware file hashes. These indicators help detect attacks with known behavior, but how well do they work for new attack campaigns or even attack variants? It is relatively easy for attackers to rotate their domain and IP infrastructures to evade static indicators. Determining a set of malware invariants to detect evolving attacks is one of the challenges in this space.

Multiple Parties. We experimented with information sharing between two networks, but can information sharing across multiple defenders provide an opportunity to develop more resilient ML detectors? Possible architectures include centralized threat sharing platforms (similar to MIPS), and peer-to-peer models. The P2P model offers a more flexible trust model, while the centralized model provides more coordination and a global perspective on the threat landscape. An important open question is how to determine which information is trustworthy and how to detect potential attackers that manipulate the shared information.

Privacy considerations. When designing threat information sharing methods, privacy of the security logs needs to be maintained. Federated learning provides decentralized learning methods to aggregate local model parameters trained by individual clients into a global ML model. Its applicability to ML models for threat detection is an interesting topic of future work.

Acknowledgements

This research was sponsored by the contract number W911NF-18-C0019 with the U.S. Army Contracting Command - Aberdeen Proving Ground (ACC-APG) and the Defense Advanced Research Projects Agency (DARPA), and by the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ACC-APG, DARPA, Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. This project was also funded by NSF under grant CNS-171763.

References

[1] MISP - Threat Intelligence Sharing Platform. https://github.com/MISP/MISP.

[2] Delivering on the promise of security AI to help defenders protect today's hybrid environments - The Official Microsoft Blog, Feb 2020. https://blogs.microsoft.com/blog/2020/02/20/delivering-on-the-promise-of-security-ai-to-help-defenders-protect-todays-hybrid-environments [accessed 11. Jun. 2020].

[3] Machine Learning: Symantec's Past, Present, and Future, Jun 2020. https://symantec-enterprise-blogs.security.com/blogs/feature-stories/machine-learning-symantecs-past-present-and-future [accessed 11. Jun. 2020].

[4] Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise, Mar 2021. [Online; accessed 3. Mar. 2021].

[5] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the Mirai botnet. In *26th USENIX Security Symposium*, pages 1093–1110, 2017.

[6] N. Boggs, S. Hiremagalore, A. Stavrou, and S. J. Stolfo. Cross-domain collaborative anomaly detection: So far yet so close. In *International Workshop on Recent Advances in Intrusion Detection*, pages 142–160. Springer, 2011.

[7] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[8] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In *Proceedings 2002 IEEE symposium on security and privacy*, pages 202–215. IEEE, 2002.

[9] L. Dandurand and O. S. Serrano. Towards improved cyber security information sharing. In *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, pages 1–16. IEEE, 2013.

[10] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank. Online adaboost-based parameterized methods for dynamic distributed network intrusion detection. *IEEE Transactions on Cybernetics*, 44(1):66–82, 2013.

[11] P. Kampanakis. Security automation and threat information-sharing options. *IEEE Security & Privacy*, 12(5):42–51, 2014.

[12] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi. DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767. IEEE, 2019.

[13] P. Ning, S. Jajodia, and X. S. Wang. Abstraction-based intrusion detection in distributed environments. *ACM Transactions on Information and System Security (TISSEC)*, 4(4):407–452, 2001.

[14] T. Ongun, O. Spohngellert, B. Miller, S. Boboila, A. Oprea, T. Eliassi-Rad, J. Hiser, A. Nottingham, J. Davidson, and M. Veeraraghavan. Portfiler: Port-level network profiling for self-propagating malware detection. `"https://www.ccs.neu.edu/home/tongun/files/TrafficProfiler.pdf"`, 2021.

[15] A. Oprea, Z. Li, R. Norris, and K. Bowers. MADE: Security analytics for enterprise threat detection. In *In 34th Annual Computer Security Applications Conference*, pages 124–136, 2018.

[16] A. Parmisano, S. Garcia, and M. J. Erquiaga. Stratosphere laboratory. a labeled dataset with malicious and benign IoT network traffic. *https://www.stratosphereips.org/datasets-iot23*, January 2019.

[17] T. Z. Project. Zeek docs. `https://docs.zeek.org/en/current/scripts/base/protocols/conn/main.zeek.html`, 2019.

[18] S. S. Response. What you need to know about the wannacry ransomware. `https://symantec-blogs.broadcom.com/blogs/threat-intelligence/wannacry-ransomware-attack`, 2017.

[19] F. Skopik, G. Settanni, and R. Fiedler. A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing. *Computers & Security*, 60:154–176, 2016.

[20] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer. Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys (CSUR)*, 47(4):1–33, 2015.

[21] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang. Machine learning and deep learning methods for cybersecurity. *Ieee access*, 6:35365–35381, 2018.

[22] J. Yang, P. Ning, X. S. Wang, and S. Jajodia. Cards:

A distributed system for detecting coordinated attacks. In *IFIP International Information Security Conference*, pages 171–180. Springer, 2000.

[23] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *In 29th Annual Computer Security Applications Conference*, pages 199–208, 2013.

A Additional Experiments.

We also conducted several experiments intended to further understand why the slow Mirai variant exhibits better recall on some ports compared to others in Figure 4b.

To this end, we aggregated the feature distances to produce a single distance metric per port. We experimented with two aggregation methods: a simple average and a weighted average where the weights are the feature importance coefficients. We analyzed whether weight sharing performs better on ports with shortest aggregated distance between the two networks.

Before aggregation, distances per feature were computed with the following methods:

- Euclidean distance over the first four moments of distribution as described in the methodology section.
- Euclidean distance over scale-adjusted moments of distribution. Given that the orders of magnitude of mean, variance, skewness and kurtosis are $E[X]$, $E[X^2]$, $E[X^3]$ and $E[X^4]$, respectively, we compute $E[X], E[X^2]^{1/2}, E[X^3]^{1/3}, E[X^4]^{1/4}$ to bring them to the same scale.
- For completeness, we also experimented with Earth mover's distance (Wasserstein metric) between feature distributions on the two networks, instead of using the moments of distribution. This method is not intended for practical uses, due to the difficulty of transferring entire feature distributions from one network to the other.

While these experiments helped us better understand the data, they were not conclusive in explaining the difference in performance on the five ports in Figure 4b. Since traffic patterns and feature distributions are quite complex, a single aggregated distance metric may not capture the necessary information. Determining on which ports the weights transfer better is more subtle and requires further research.

B Comparison of Sharing Methods.

We compare the three methods proposed using various performance metrics. We have previously discussed the recall metric in Section 3. In this section, we look at precision and false positives for each of the five ports on the slow Mirai variant.

	Infected Port				
	80	**443**	**22**	**23**	**445**
Model Sharing	0.16	0	0.06	0.86	0.03
Weight Sharing	0.35	0.61	0.06	0.70	0.70
Weight Adaptation	1.0	0.81	0.96	0.98	0.76

Table 1: Performance in terms of Precision in the top-60 ranked alerts across the ports, for the slow Mirai variant at VT.

	Infected Port				
	80	**443**	**22**	**23**	**445**
Model Sharing	50	60	56	8	58
Weight Sharing	39	23	56	18	18
Weight Adaptation	0	11	2	1	14

Table 2: Performance in terms of False Positives in the top-60 ranked alerts across the ports, for the slow Mirai variant at VT.

Table 1 illustrates performance in terms of precision, while Table 2 illustrates performance in terms of false positives in top-60 alerts.

As these tables show, the Weight Sharing method generally improves on the Model Sharing method. The Weight Adaptation method delivers best precision with fewest false positives across the board.

C PORTFILER: Traffic Features.

We present a high-level overview of features used in PortFiler in four categories. We extract these features for each port separately. We define these features for each 1-minute time window. The complete list of features are described in [14].

1. **Traffic statistics features:** We extract the number of distinct internal and external IPs communicating on that port, number of connections, and number of new distinct external IPs.

2. **Duration features**: We extract max, min, variance and mean of duration values.

3. **Bytes and packets features**: We extract max, variance, and mean of sent and received bytes and packets values. We also define the number of connections with no bytes received as a separate feature.

4. **Connection state features:** We extract the number of connections for each Zeek connection state string (e.g., S0, S1, OTH). We also define the number of failed connections as a separate feature.