

Safeguarding Network Intrusion Detection Models from Zero-day Attacks and Concept Drift

Brian Matejek¹, Ashish Gehani¹, Nathaniel D. Bastian², Daniel Clouse³, Bradford Kline³, Susmit Jha¹

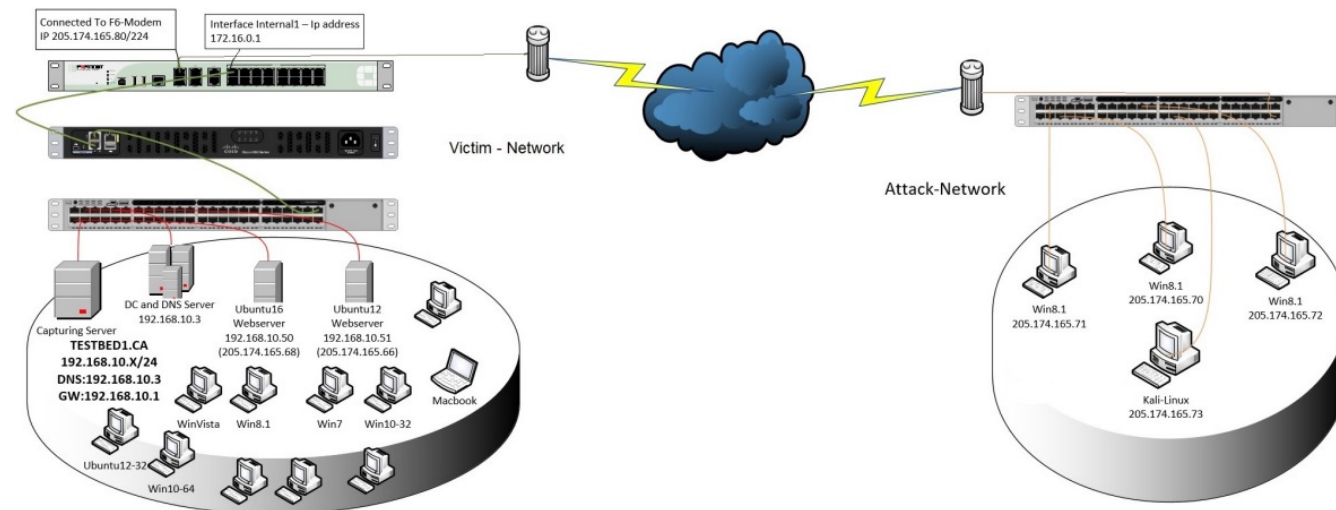
¹Computer Science Laboratory, SRI International

²Army Cyber Institute, United States Military Academy

³Laboratory for Advanced Cybersecurity Research, Department of Defense

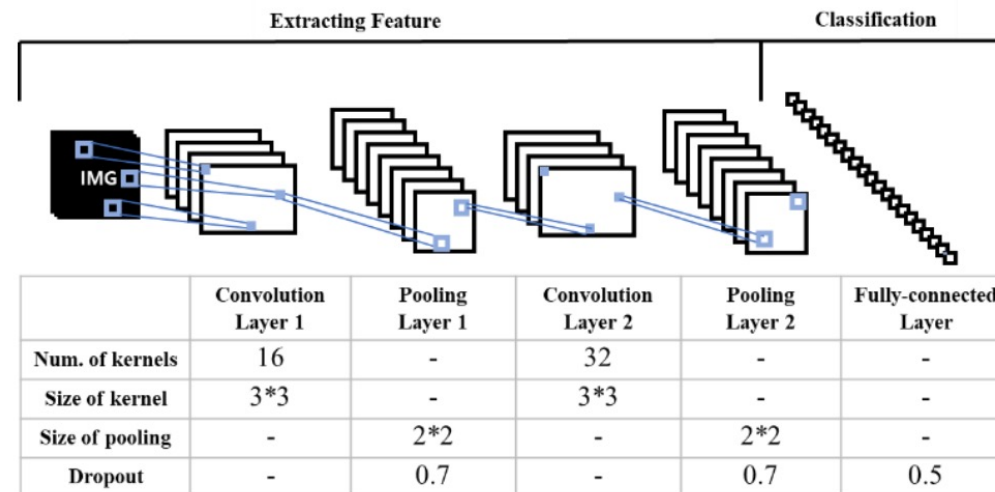
Network Intrusion Detection Systems

- Network intrusion detection systems (NIDS) monitor network traffic to identify potentially malicious or anomalous activity.
- Increasingly, these systems have relied on machine learning to increase detection speed and reduce the cognitive load on cybersecurity operators.



Machine Learning for Intrusion Detection

- A significant amount of existing literature on machine learning for NIDS focuses on hand-engineered features at the *flow-level*.
- These features might include information on the average payload length, the number of forward and backwards TCP packets with certain flags, or the flow duration.
- The flow features can either serve as input into a classical ML model such as random forest, or into deep learning models such as a fully-connected neural network or convolutional neural network.



Packet-Level versus Flow-Level Detection



- We focus on *packet-level* detection of malicious activity.
- There are two main benefits for analyzing traffic at the packet-level:
 - First, the configuration of flow capture devices varies across different networks and operators, creating difficulty in constructing an ML solution that works across a wide range of networks.
 - Second, flow-level detection is inherently retroactive. The flow must complete before inference can occur.
- Existing tools and libraries such as tcpdump and libpcap allow for the capture of all network traffic at the packet-level and work across a wide range of operating systems.



Out-of-Distribution Inputs



- Machine learning models perform exceptionally well when inferring on samples **within** the training distribution.
- However, these networks fail to generalize on novel inputs that **fall outside** the training distribution.
- Worse still, these networks are often over-confident on these inputs.

Out-of-Distribution in Cybersecurity

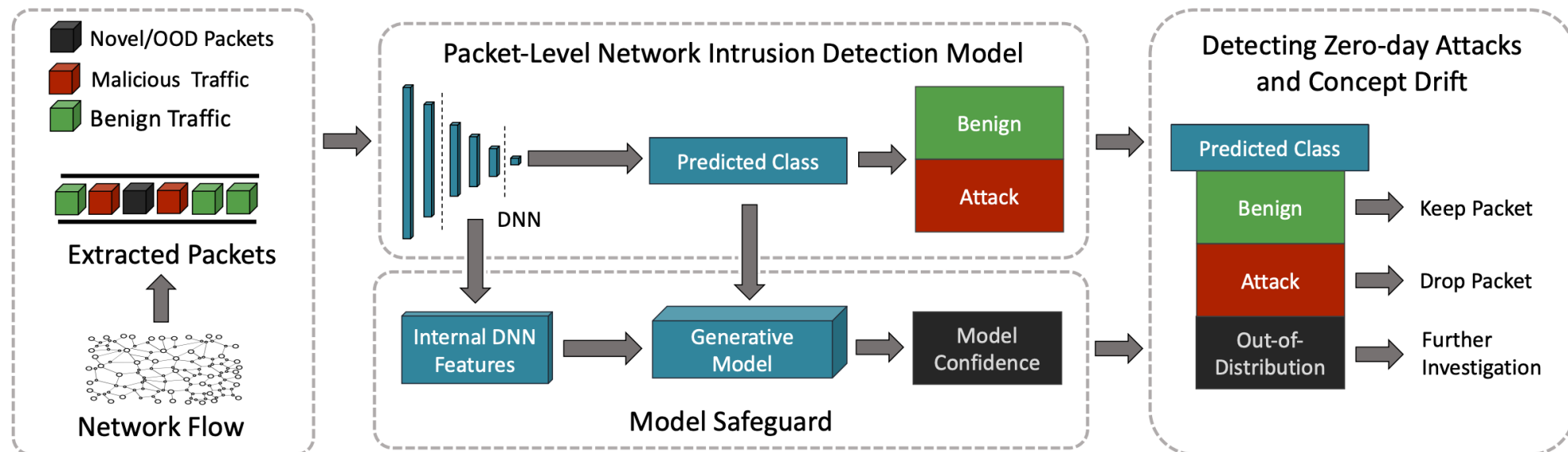


- We consider two sources of out-of-distribution (OOD) data in network traffic.
- Zero-day attacks refer to novel exploits previously unseen.
 - These attacks cannot, by definition, appear in the training data.
- Concept drift refers to evolving network traffic as the structure of networks and nature of activity change.
 - For example, a surge in teleconferencing activity since March 2020, a shift to encrypted protocols, or increased connections to Internet-of-Things devices.

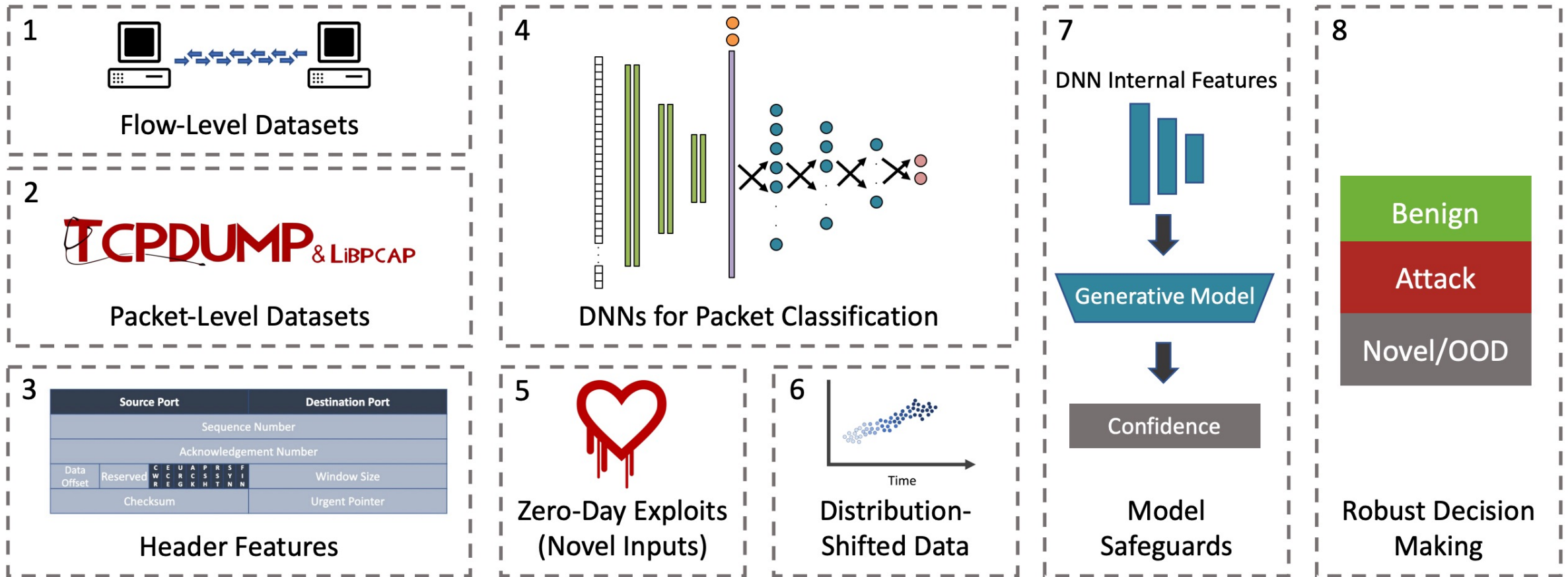
Proposed Framework



- We propose a network intrusion detection model with a model safeguard that ingests the raw payloads from network traffic to predict benign or malicious activity.
- The model safeguard provides a confidence score for the discriminative classifier to detect OOD inputs such as zero-day attacks and shifting distributions of traffic.
 - Samples with a low confidence score from the model safeguard require further (human) investigation.



Methodology



1



Flow-Level Datasets

2



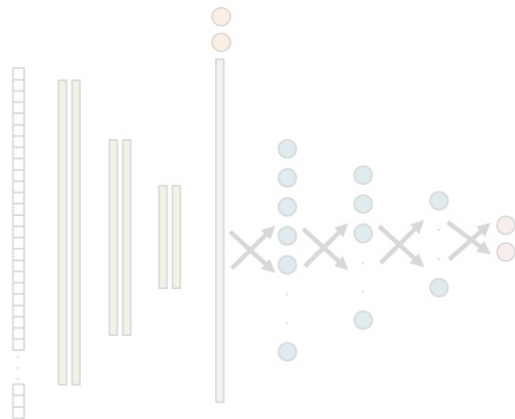
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4



DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

6



Distribution-Shifted Data

7

DNN Internal Features



Generative Model

Confidence

Model
Safeguards

8

Benign

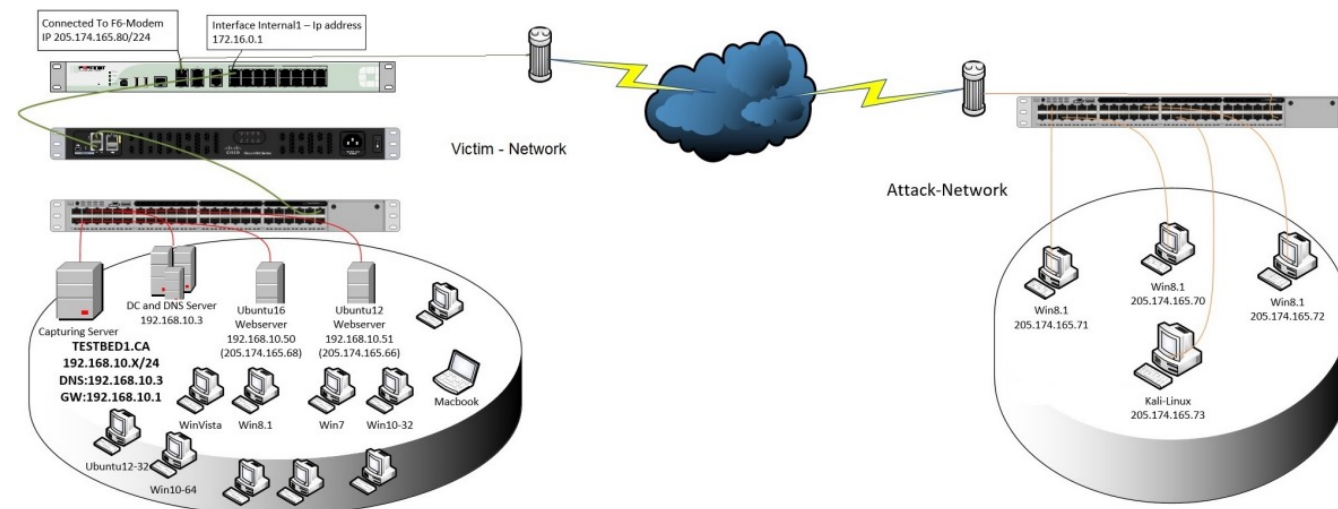
Attack

Novel/OOD

Robust Decision
Making

Flow-Level Datasets

- We repurpose existing flow-level datasets such as UNSW-NB15 and CIC-IDS-2017.
- These academic datasets typically have victim and attack computers with set IP addresses and monitor traffic through a single point of entry.
- The attack network uses several different exploits over the course of a few days or weeks.



1



Flow-Level Datasets

2

TCPDUMP & **LIBPCAP**

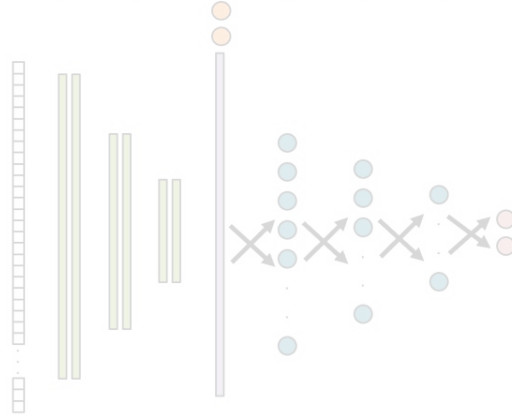
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4



DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

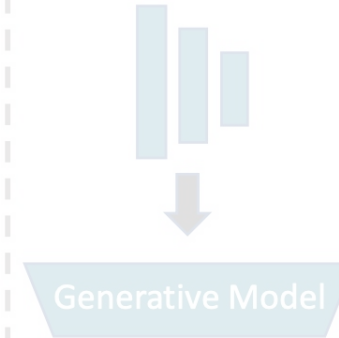
6



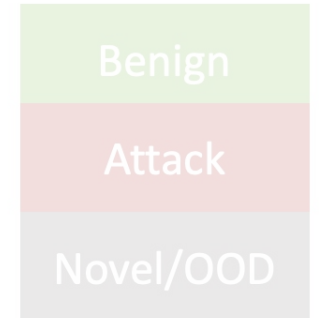
Distribution-Shifted Data

7

DNN Internal Features

Model
Safeguards

8

Robust Decision
Making

- We convert the existing flow-level datasets into packet-level ones by enumerating all TCP and UDP packets in the PCAP files and assigning them category labels.
- Since many flow capture systems are bidirectional, for each packet we extract its five-tuple and its reciprocal:
(src_ip, src_port, dest_ip, dest_port, protocol) **AND** (dest_ip, dest_port, src_ip, src_port, protocol)
- We identify candidate flows based on these keys and narrow our selection based on timing information.
 - Many academic datasets reuse IP addresses and ports for multiple flows; thus, timing information is critical for proper labeling.
 - **N.B.:** The packets often have finer precision in their timestamps than the captured flows, so one must be careful to identify the correct flow.

Payload Processing



- We truncate payloads to 1500 bytes to match the typical maximum transmission of ethernet networks.
- Payloads with fewer than 1500 bytes are zero padded.

1



Flow-Level Datasets

2

TCPDUMP & LIBPCAP

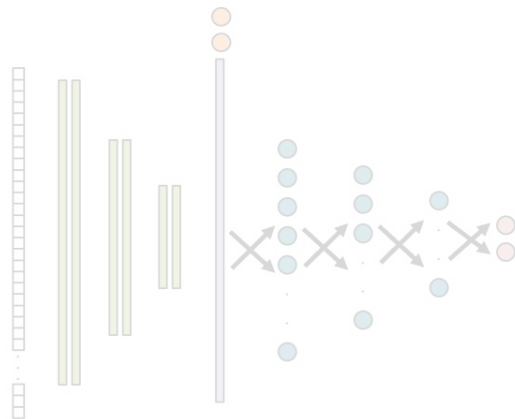
Packet-Level Datasets

3

Source Port			Destination Port		
Sequence Number					
Acknowledgement Number					
Data Offset	Reserved	C W R	E C E G	A C S K	P S S H
Checksum			Window Size		
			Urgent Pointer		

Header Features

4



DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

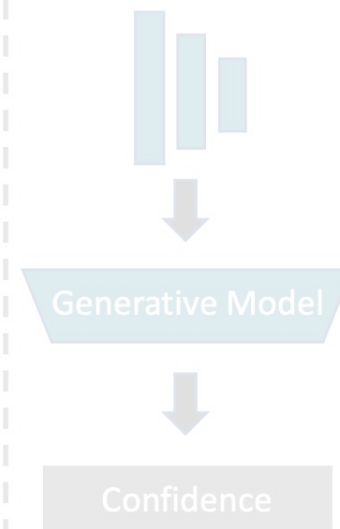
6



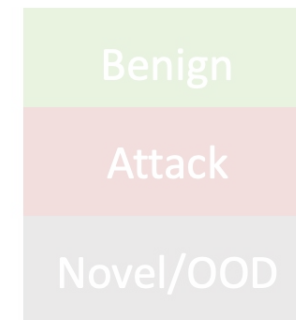
Distribution-Shifted Data

7

DNN Internal Features

Model
Safeguards

8

Robust Decision
Making

Header Features



- We cannot simply provide the entire header to our neural network as the model would learn a trivial mapping from the input space to the packet label.
 - For example, network intrusion detection datasets use set IP addresses for the attack and victim computers.
- We include Total Length from IPv4 header, modified Source Port and Destination Port from TCP and UDP headers, and the values of eight flags in TCP: CWR, ECE, URG, ACK, PSH, RST, SYN, and FIN.

IPv4 Header

Version	IHL	TOS	Total Length	
Identification			Flags	Fragment Offset
TTL		Protocol	Header Checksum	
Source Address				
Destination Address				

TCP Header

Source Port		Destination Port									
Sequence Number											
Acknowledgement Number											
Data Offset	Reserved	C W R E	E C R E	U R G	A C K	P S H	R S S	S S	F I N	Window Size	
Checksum						Urgent Pointer					

UDP Header

Source Port		Destination Port	
Length		Checksum	

Source and Destination Port Features



- We create a binary vector to encode source and destination port information based on existing protocols and port groupings.
- This is not a one-hot-encoding as certain port numbers fall in two bins (e.g., port 8080 for HTTP and Registered Ports).
- We need to group in this manner to avoid learning the dataset configuration instead of the signal in the payloads themselves.

Port Number	Description
21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
80/8080	Hypertext Transfer Protocol (HTTP)
443/444	Hypertext Transfer Protocol Secure (HTTPS)
0 - 1,023	Well Known Ports
1,024 - 49,151	Registered Ports
49,152 - 65535	Dynamic/Private Ports

1



Flow-Level Datasets

2



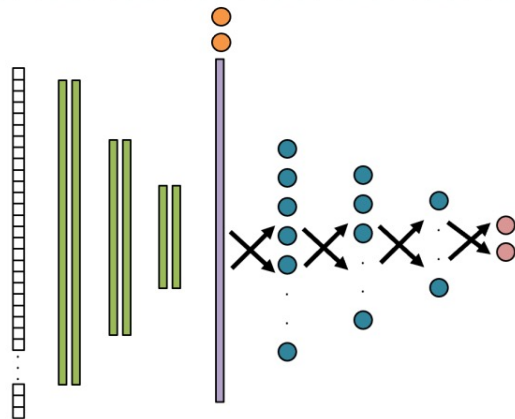
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4



DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

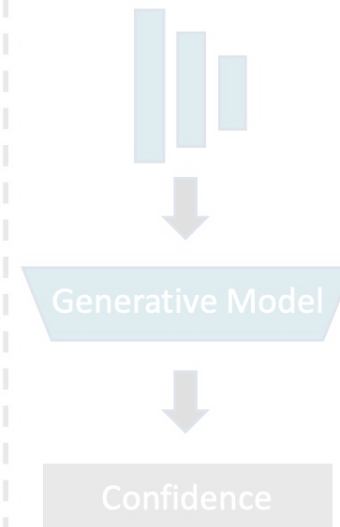
6



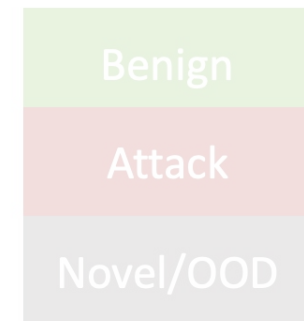
Distribution-Shifted Data

7

DNN Internal Features

Model
Safeguards

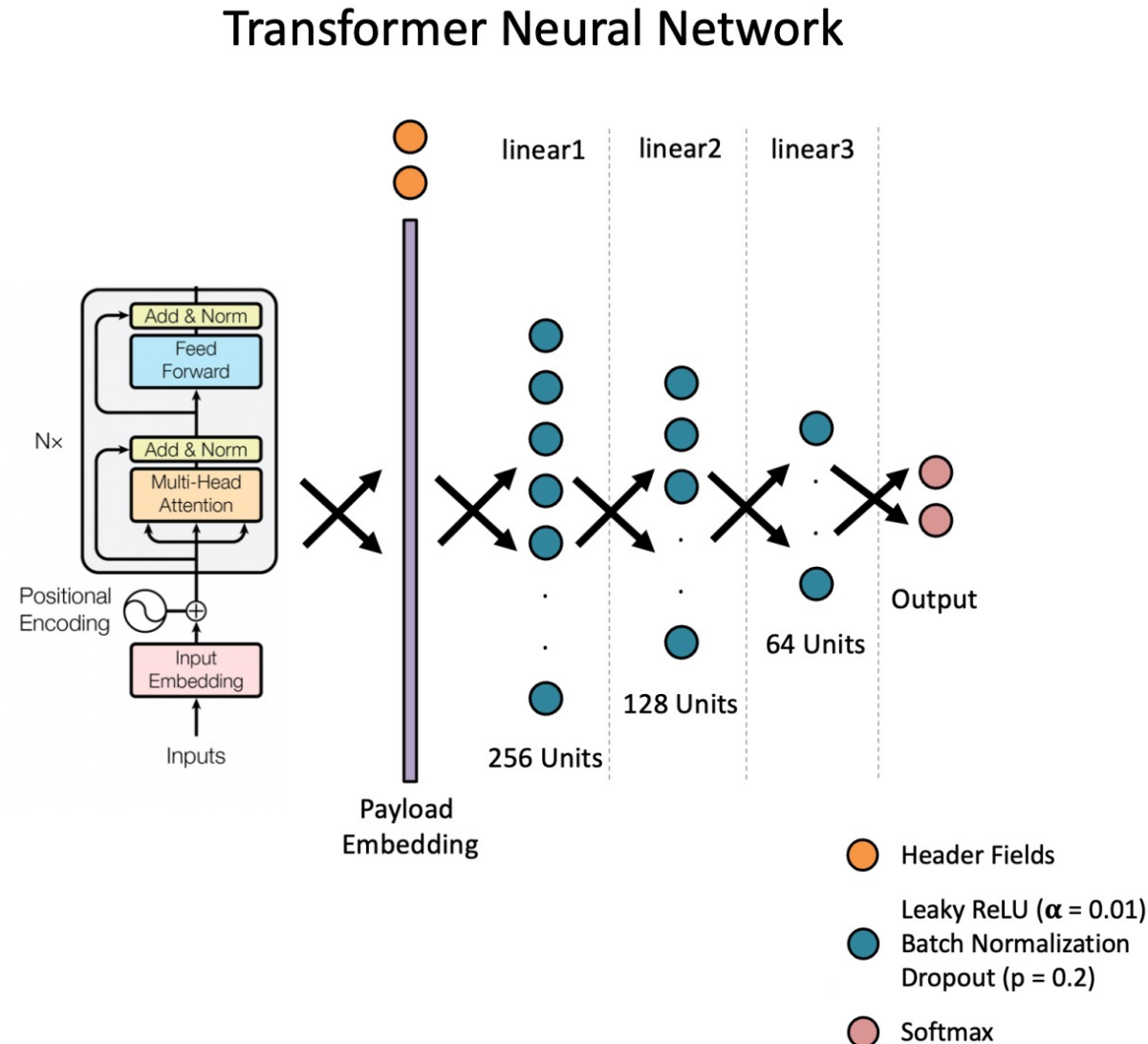
8

Robust Decision
Making

DNNs For Packet Classification



- We use an encoder-only transformer network for the binary malicious/benign packet classification task.
- We stack two transformer blocks with an embedding size of 384 and six self-attention heads for 64-dimensional key, value, and query vectors.
- We use mean pooling to convert the output from the last transformer block into a sentence or "payload" embedding.
- After concatenating the header features to the embedding, we have three fully-connected with 256, 128, and 64 neurons.

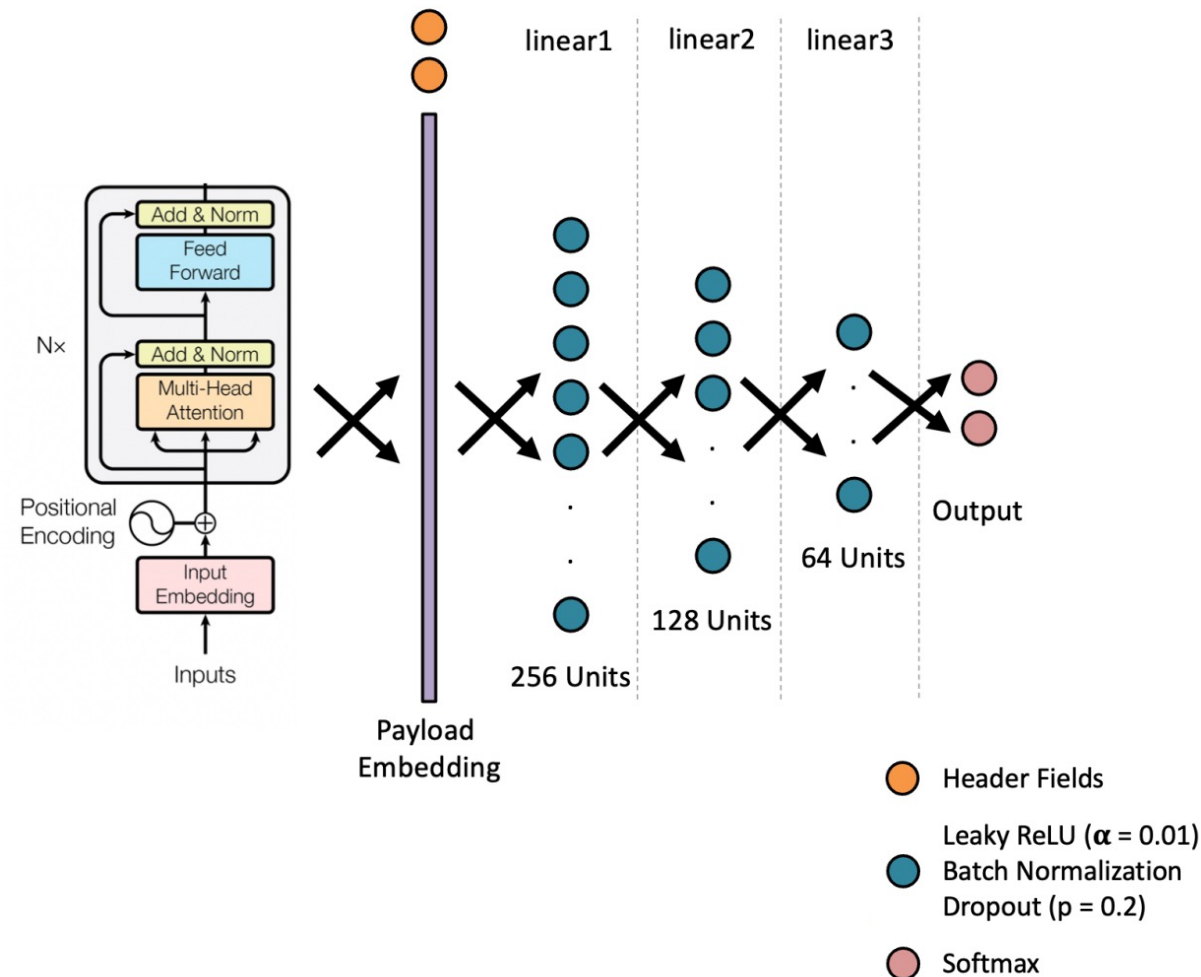


Custom Tokenization



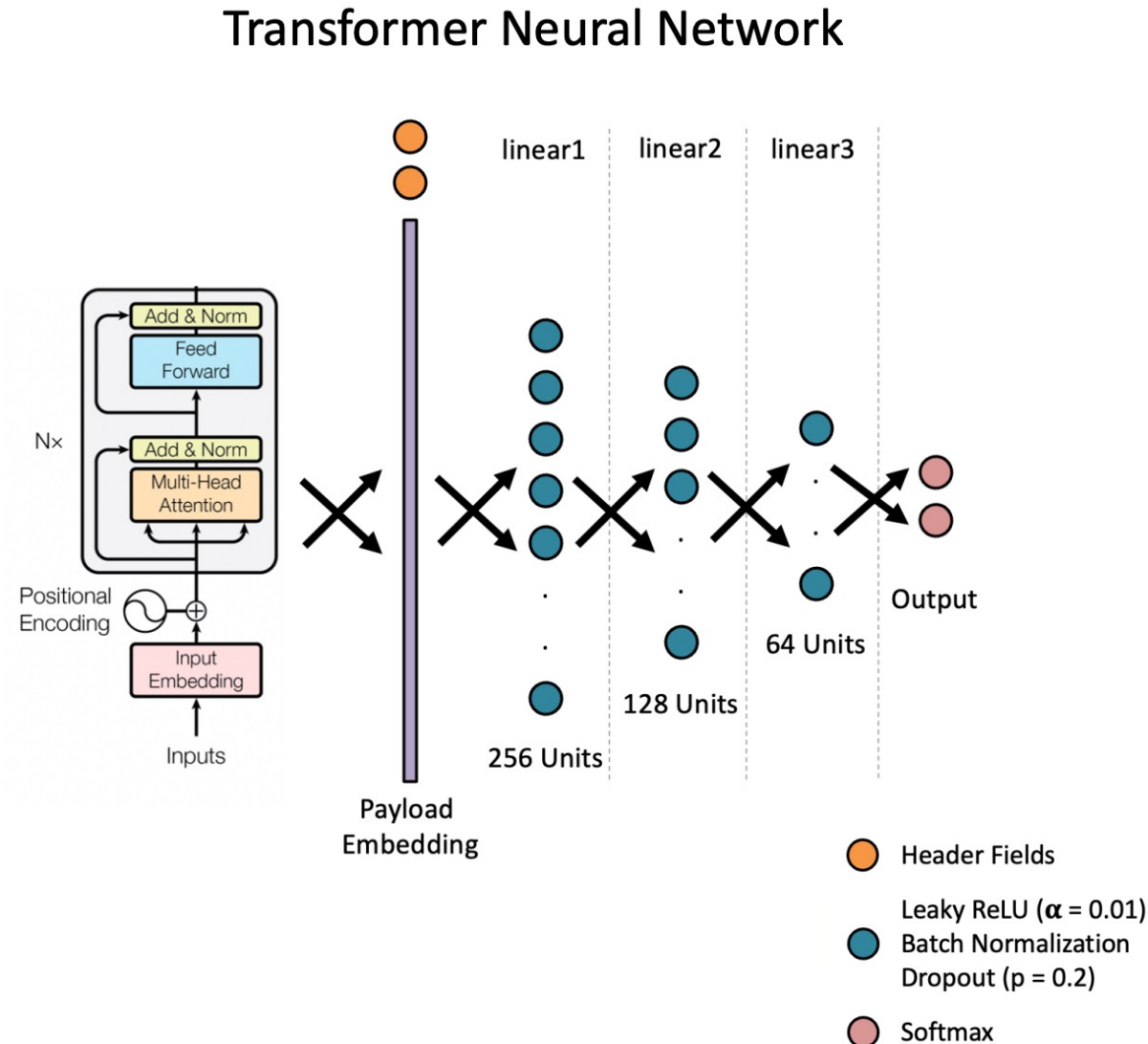
- We create a custom, simple tokenization strategy to convert the payloads into tokens for input.
- Each unique byte from 0 to 255 receives that corresponding token value.
 - We have a vocabular size of 256 words.
- We have found that existing tokenization strategies tuned for natural languages provide sub-optimal results on the high entropy inputs of the packet-level data.

Transformer Neural Network



Fully-Connected Layers

- We use leaky ReLU activations on the fully-connected layers with $\alpha = 0.01$, followed by batch normalization and dropout.
- The final output layers use a softmax activation function.



1



Flow-Level Datasets

2

TCPDUMP & LIBPCAP

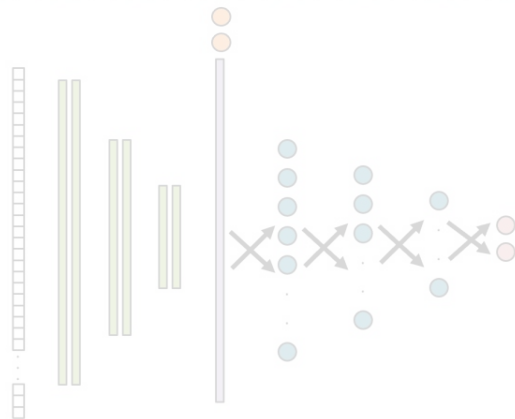
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4



DNNs for Packet Classification

5



Zero-Day Exploits
(Novel Inputs)

6



Distribution-
Shifted Data

7

DNN Internal Features



Generative Model

Confidence

Model
Safeguards

8

Benign

Attack

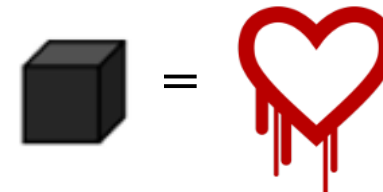
Novel/OOD

Robust Decision
Making

Zero-day Exploits (Novel Inputs)



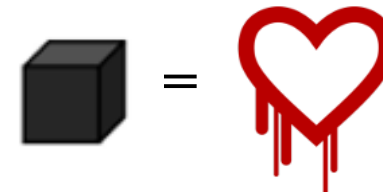
- Machine learning models generally perform well on in-distribution data and generalize poorly on out-of-distribution data.
- Perhaps worse, these ML models are often over-confident on out-of-distribution data making the output values (logits etc.) unreliable.
- We model the introduction of a novel exploit into the world by training models on all but one of the attack categories.
 - That is, one model may contain all benign and malicious training samples except for those corresponding to a Heartbleed attack.



Zero-day Exploits (Novel Inputs)



- During inference, we provide all benign and malicious testing samples, including those from the excluded class.
 - Some attacks are similar in patterns to others, e.g., the various variants of distributed denial-of-service exploits.
 - Others, however, represent truly novel activity and are incorrectly labeled benign with high confidence by our networks.



1



Flow-Level Datasets

2

TCPDUMP & LIBPCAP

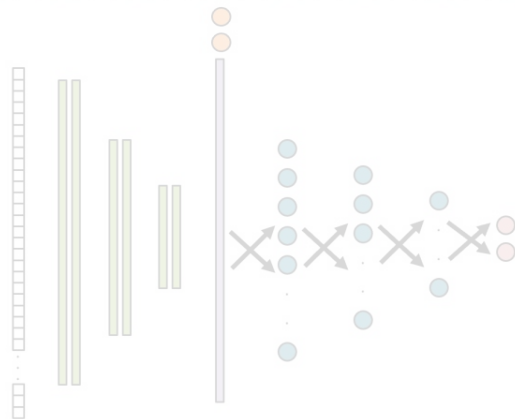
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4

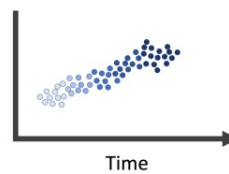


DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

6



Distribution-Shifted Data

7

DNN Internal Features



Generative Model

Confidence

Model
Safeguards

8

Benign

Attack

Novel/OOD

Robust Decision
Making

Distribution-Shifted Data



- Internet best-practices evolve over time such as the gradual shift from unencrypted to encrypted communications (e.g., HTTP to HTTPS).
- We model such concept drift by training on data from one year and inferring on data from another year.
- The datasets we use reflect the rapid move from HTTP in 2015 to HTTPS by late 2017.

Port(s)	CIC-IDS-2017		UNSW-NB15	
Source Port	No. Benign	No. Attack	No. Benign	No. Attack
21	40,417	17,975	689,008	19,288
22	128,195	43,651	1,592,854	284
80/8080	13,912,273	757,410	15,021,418	450,780
443/444	7,916,684	19,020	0	268
Well Known (0 - 1023)	24,025,879	837,249	19,186,346	1,015,630
Registered (1024 - 49151)	1,436,391	166,559	28,350,557	1,098,046
Dynamic/Private (49152 - 65535)	3,722,886	216,220	2,217,170	344,656
Destination Port	No. Benign	No. Attack	No. Benign	No. Attack
21	30,470	17,947	500,000	17,192
22	124,003	46,594	1,187,386	314
80/8080	589,520	286,401	354,910	112,911
443/444	2,305,603	31,026	0	394
Well Known (0 - 1023)	5,197,451	380,195	4,366,251	1,384,884
Registered (1024 - 49151)	3,618,349	440,957	34,715,404	789,883
Dynamic/Private (49152 - 65535)	20,369,356	398,876	10,672,418	283,565

1



Flow-Level Datasets

2



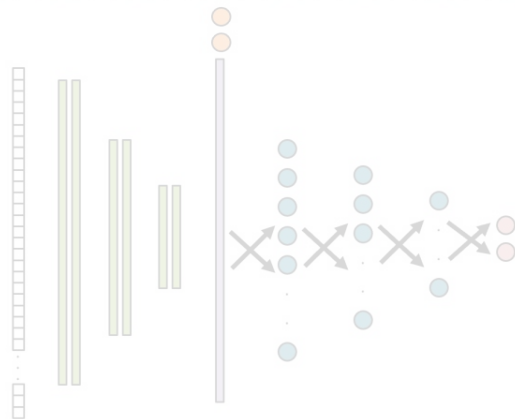
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4



DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

6



Distribution-Shifted Data

7

DNN Internal Features



Generative Model

Confidence

Model
Safeguards

8

Benign

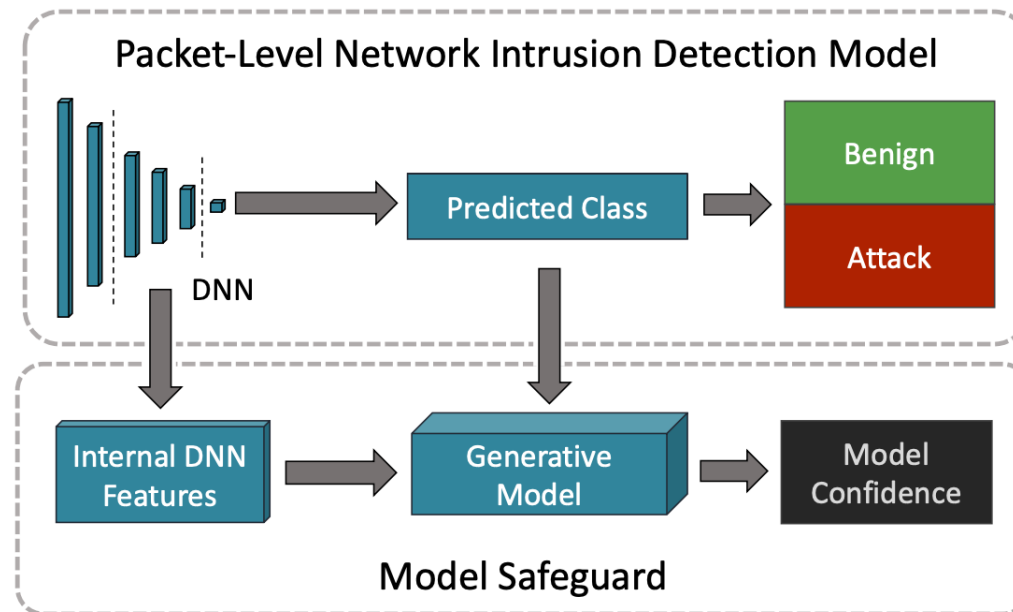
Attack

Novel/OOD

Robust Decision
Making

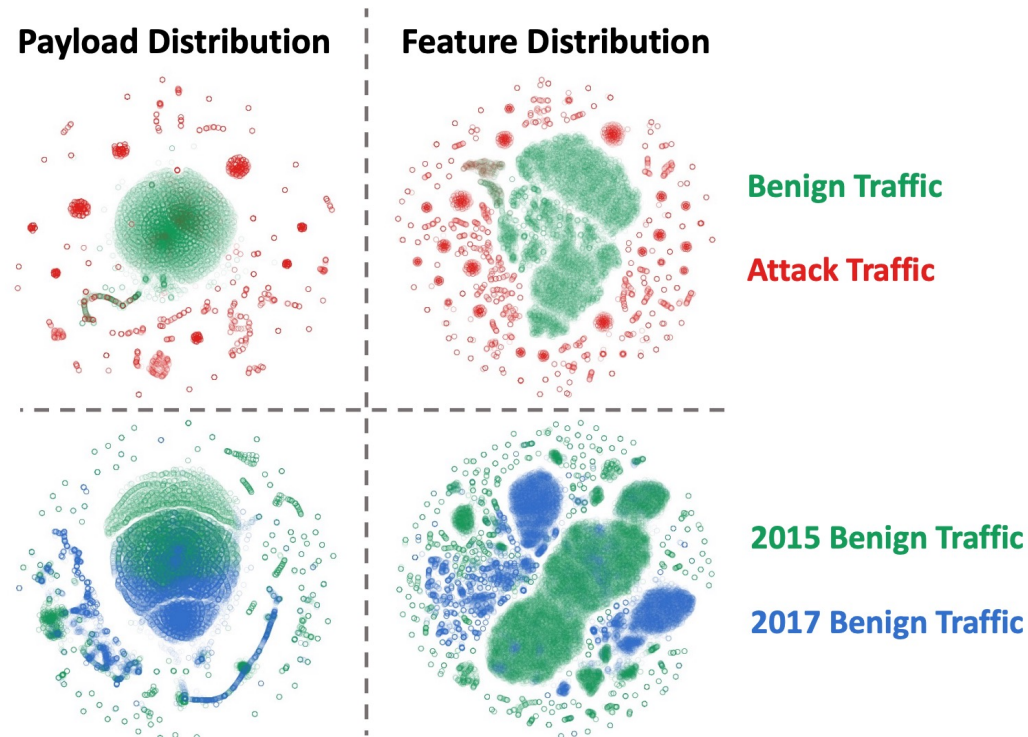
DNN Internal Features

- We extract the outputs from the intermediate fully-connected layers from our discriminative classifier to feed as input into our generative model.
- This model safeguard learns the expected distributions of these extracted layers for data that comes from the same distribution as the training data.
- Inputs that produce features outside of the expected distribution are flagged as out-of-distribution, and thus our discriminative model's prediction untrustworthy.



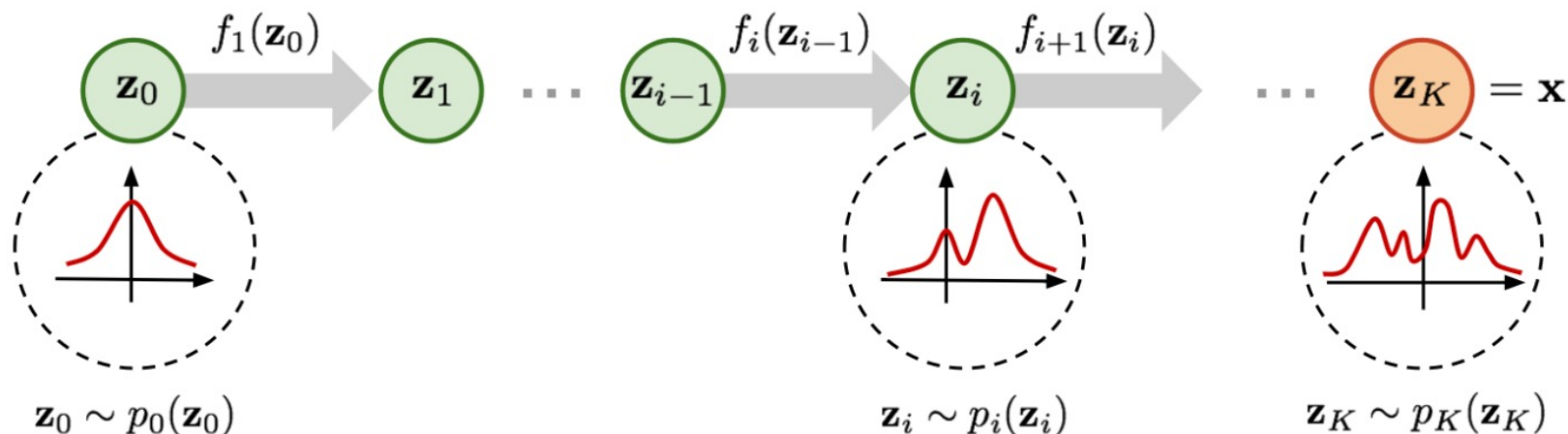
Why Use the DNN Internal Features?

- The intermediate features better separate the classes than the payloads themselves.
 - t-SNE plots below show the separation in the feature space compared to the input (payload) space.
- The final output layer is poorly calibrated with little differentiable between in-distribution and out-of-distribution examples.



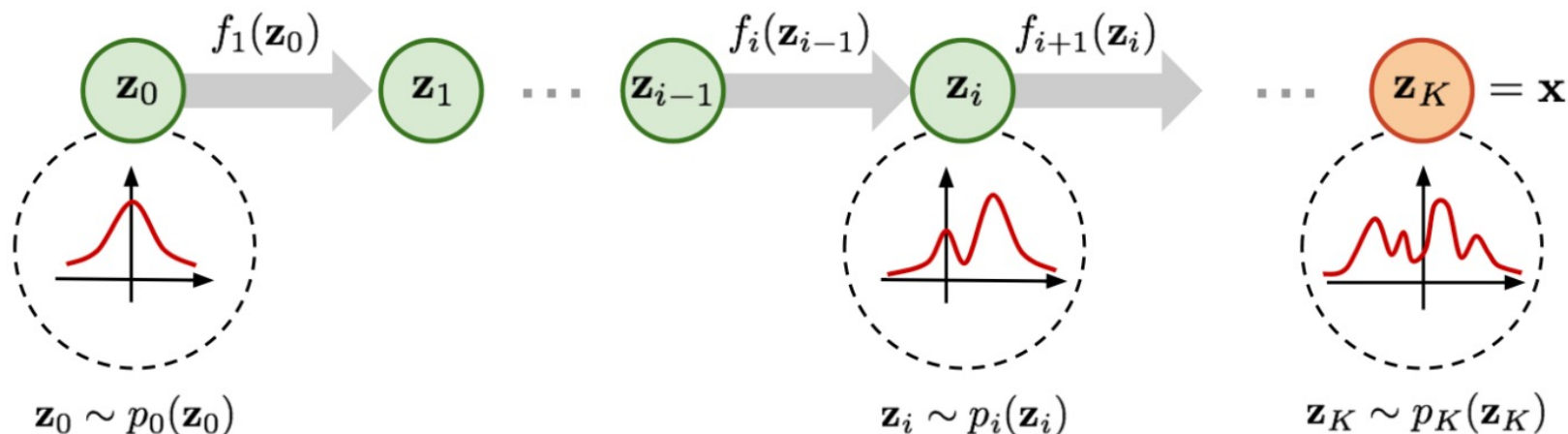
Normalizing Flows

- Normalizing flows use a series of bijective transformations to create a one-to-one mapping from a simple distribution, such as a multivariate Gaussian, to a complex one, such as the internal feature space of our discriminative classifier.
- We train class-conditional normalizing flows (i.e., one for benign payloads and one for malicious payloads).
- We add a small amount of Gaussian noise to our extracted features $\mathcal{N}(0, 0.05)$ to wash out non-informative features with small variances.



Normalizing Flows

- During training, we sample from $\mathbf{z}_0$ (multivariate Gaussian) and learn the intermediate $\mathbf{z}_i$ to match $\mathbf{x}$, the extracted features from our in-distribution training data.
- During inference, we have $\mathbf{x}$ for all extracted features and can get the corresponding sample in $\mathbf{z}_0$ for each.
- Since $\mathbf{z}_0$ is a simple distribution, we can generate relative likelihoods of each sample belonging to in-distribution data.



1



Flow-Level Datasets

2

TCPDUMP & LIBPCAP

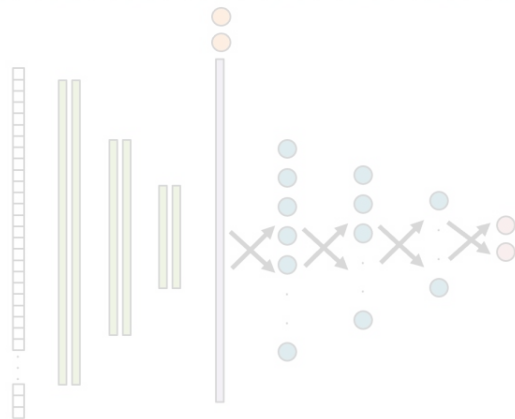
Packet-Level Datasets

3

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
Data Offset	Reserved	C E U A P R S F W C R C S S Y I R E G K H T N N	Window Size
Checksum		Urgent Pointer	

Header Features

4



DNNs for Packet Classification

5

Zero-Day Exploits
(Novel Inputs)

6



Distribution-Shifted Data

7

DNN Internal Features



Generative Model

Confidence

Model
Safeguards

8

Benign

Attack

Novel/OOD

Robust Decision
Making

Robust Decision Making



- Our models work in tandem to produce a label for each packet as well as a confidence score.
- The discriminative model produces a binary label of benign or malicious.
- The class-conditional generative models produce a confidence that the input to the discriminative model is in-distribution and therefore the discriminative model's prediction trustworthy.
- Inputs that deviate significantly from the learned distribution are flagged as out-of-distribution and require further (human) analysis.

Experimental Setup



- We use the CIC-IDS-2017 and UNSW-NB15 datasets for our experiments.
- We create an equal balance of benign and malicious samples and randomly sample the data stratified by packet category to create ten training, validation, and testing splits.

CIC-IDS-2017

Packet Category	No. Packets	Avg. Payload Length
Benign	29,185,156	3,017.95 ($\pm 2,444.65$)
Bot	2,584	4,088.69 ($\pm 6,859.15$)
DDoS	220,160	6,166.16 ($\pm 5,822.03$)
DoS GoldenEye	26,198	5,484.42 ($\pm 5,509.26$)
DoS Hulk	747,340	4,810.35 ($\pm 4,467.81$)
DoS Slowhttptest	5,906	1,049.66 (± 965.19)
DoS Slowloris	23,593	404.80 (± 204.47)
FTP-Patator	35,922	44.06 (± 14.41)
Heartbleed	20,181	7,665.59 ($\pm 4,815.72$)
Infiltration	29,865	908.23 (± 551.25)
Port Scan	417	4,761.18 ($\pm 3,781.50$)
SSH-Patator	90,245	307.56 (± 454.89)
Web Attack-Brute Force	14,400	1,148.79 (± 590.99)
Web Attack-SQL Injection	19	2,511.68 ($\pm 1,966.98$)
Web Attack-XSS	3,198	2,280.02 ($\pm 1,398.51$)

UNSW-NB15

Packet Category	No. Packets	Avg. Payload Length
Analysis	1,394	561.35 (± 273.80)
Backdoor	1,246	733.00 (± 929.74)
Benign	49,754,073	2,140.15 ($\pm 1,195.68$)
DoS	479,275	2,326.59 ($\pm 1,026.94$)
Exploits	1,619,721	2,250.03 ($\pm 1,148.98$)
Fuzzers	144,114	1,688.62 ($\pm 1,330.87$)
Generic	192,901	2,429.40 (± 971.32)
Reconnaissance	9,601	189.90 (± 140.33)
Shellcode	828	271.03 (± 229.82)
Worms	9,252	2,575.79 (± 851.72)

Binary Classification Results



- When inferring only in-distribution data, the discriminative classifier achieves exceptionally high accuracies.
- We note that the number of false positives is significantly lower than the number of false negatives (i.e., we are more likely to erroneously label a malicious packet as benign than vice versa).

Metric	Result (↑)
Accuracy	99.40%
AU ROC	0.9997
F1-Score	0.9940

Modeling Zero-day Attacks



- The in-distribution column shows the proportion of attack packets for a given attack category that are correctly classified by our discriminative model **when included** in the training distribution.

Attack Category	No. Test Packets	In-Distribution ($\uparrow$)	Out-of-Distribution ($\uparrow$)
Bot	1,292	0.9424 (± 0.0270)	0.0950 (± 0.0935)
DDoS	110,080	1.0000 (± 0.0000)	0.2145 (± 0.0258)
DoSGoldenEye	13,099	0.9999 (± 0.0001)	0.1995 (± 0.0426)
DoSHulk	373,670	1.0000 (± 0.0000)	0.7529 (± 0.1626)
DoSSlowhttpstest	2,953	0.9980 (± 0.0025)	0.5248 (± 0.1659)
DoSSlowloris	11,796	0.9996 (± 0.0002)	0.1737 (± 0.2859)
FTP-Patator	17,961	0.9994 (± 0.0002)	0.0030 (± 0.0019)
Heartbleed	10,090	0.4418 (± 0.0660)	0.0002 (± 0.0002)
Infiltration	14,932	0.9946 (± 0.0051)	0.0780 (± 0.0312)
PortScan	208	0.9856 (± 0.0071)	0.8058 (± 0.0234)
SSH-Patator	45,123	0.9998 (± 0.0001)	0.0027 (± 0.0026)
WebAttack-BruteForce	7,200	0.9997 (± 0.0001)	0.9439 (± 0.0701)
WebAttack-SQLInjection	9	0.7222 (± 0.1745)	0.4678 (± 0.1382)
WebAttack-XSS	15,99	0.9959 (± 0.0030)	0.6305 (± 0.1717)

All rows correspond to the same trained models. Models were given the complete training data (i.e., benign data and all fourteen types of attacks).

Modeling Zero-day Attacks



- The out-of-distribution column shows the proportion of attack packets for a given attack category that are correctly classified by our discriminative model **when NOT included** in the training distribution.

Attack Category	No. Test Packets	In-Distribution ($\uparrow$)	Out-of-Distribution ($\uparrow$)
Bot	1,292	0.9424 (± 0.0270)	0.0950 (± 0.0935)
DDoS	110,080	1.0000 (± 0.0000)	0.2145 (± 0.0258)
DoSGoldenEye	13,099	0.9999 (± 0.0001)	0.1995 (± 0.0426)
DoSHulk	373,670	1.0000 (± 0.0000)	0.7529 (± 0.1626)
DoSSlowhttpstest	2,953	0.9980 (± 0.0025)	0.5248 (± 0.1659)
DoSSlowloris	11,796	0.9996 (± 0.0002)	0.1737 (± 0.2859)
FTP-Patator	17,961	0.9994 (± 0.0002)	0.0030 (± 0.0019)
Heartbleed	10,090	0.4418 (± 0.0660)	0.0002 (± 0.0002)
Infiltration	14,932	0.9946 (± 0.0051)	0.0780 (± 0.0312)
PortScan	208	0.9856 (± 0.0071)	0.8058 (± 0.0234)
SSH-Patator	45,123	0.9998 (± 0.0001)	0.0027 (± 0.0026)
WebAttack-BruteForce	7,200	0.9997 (± 0.0001)	0.9439 (± 0.0701)
WebAttack-SQLInjection	9	0.7222 (± 0.1745)	0.4678 (± 0.1382)
WebAttack-XSS	15,99	0.9959 (± 0.0030)	0.6305 (± 0.1717)

Each row is a separately trained model where we omit one attack category from the data (i.e., model in DDoS row was not trained on DDoS packets).

Modeling Zero-day Attacks



- Note the significant drops in classification accuracy for these packets when removed from the training distribution!

Attack Category	No. Test Packets	In-Distribution ($\uparrow$)	Out-of-Distribution ($\uparrow$)
Bot	1,292	0.9424 (± 0.0270)	0.0950 (± 0.0935)
DDoS	110,080	1.0000 (± 0.0000)	0.2145 (± 0.0258)
DoSGoldenEye	13,099	0.9999 (± 0.0001)	0.1995 (± 0.0426)
DoSHulk	373,670	1.0000 (± 0.0000)	0.7529 (± 0.1626)
DoSSlowhttpstest	2,953	0.9980 (± 0.0025)	0.5248 (± 0.1659)
DoSSlowloris	11,796	0.9996 (± 0.0002)	0.1737 (± 0.2859)
FTP-Patator	17,961	0.9994 (± 0.0002)	0.0030 (± 0.0019)
Heartbleed	10,090	0.4418 (± 0.0660)	0.0002 (± 0.0002)
Infiltration	14,932	0.9946 (± 0.0051)	0.0780 (± 0.0312)
PortScan	208	0.9856 (± 0.0071)	0.8058 (± 0.0234)
SSH-Patator	45,123	0.9998 (± 0.0001)	0.0027 (± 0.0026)
WebAttack-BruteForce	7,200	0.9997 (± 0.0001)	0.9439 (± 0.0701)
WebAttack-SQLInjection	9	0.7222 (± 0.1745)	0.4678 (± 0.1382)
WebAttack-XSS	15,99	0.9959 (± 0.0030)	0.6305 (± 0.1717)

When in-distribution, our model classifies 99.98% of SSH-Patator packets as malicious.
When out-of-distribution, it classifies only 0.27% of SSH-Patator packets as malicious.

Modeling Concept Drift

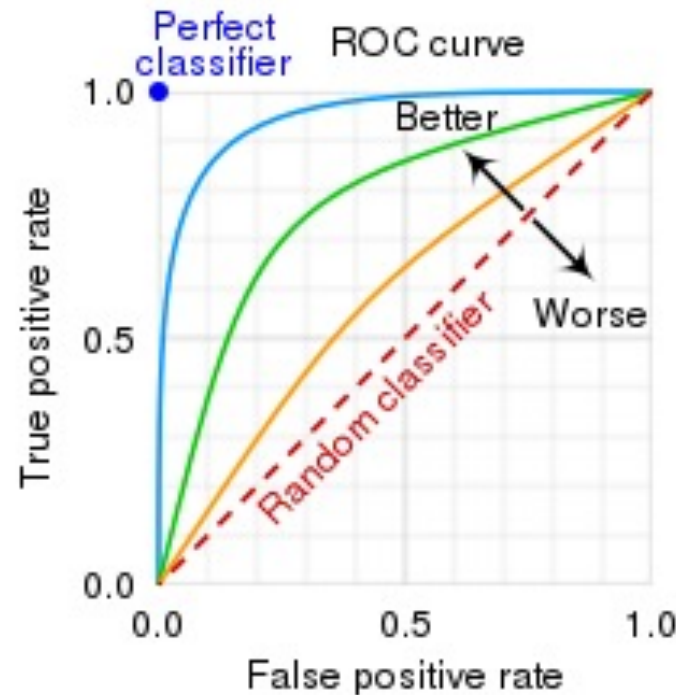


- We train our models on the CIC-IDS-2017 dataset and infer on the UNSW-NB15 one.
 - The UNSW-NB15 dataset has significantly less encrypted traffic than the CIC-IDS-2017 data as it typically uses the HTTP protocol instead of HTTPS.
- Our discriminative classifier performs poorly on the UNSW-NB15 data, classifying most malicious traffic as benign and incorrectly classifying approximately 8% of benign traffic.

Packet Category	Accuracy ($\uparrow$)
Analysis	42.24% ($\pm 6.53\%$)
Backdoor	44.19% ($\pm 3.22\%$)
Benign	92.16% ($\pm 0.43\%$)
DoS	15.29% ($\pm 1.77\%$)
Exploits	12.57% ($\pm 0.90\%$)
Fuzzers	8.12% ($\pm 0.89\%$)
Generic	5.10% ($\pm 0.40\%$)
Reconnaissance	15.81% ($\pm 2.53\%$)
Shellcode	24.81% ($\pm 2.33\%$)
Worms	1.92% ($\pm 0.39\%$)
Aggregate	48.07% ($\pm 1.34\%$)

Out-of-Distribution Metrics

- We evaluate our method using two out-of-distribution metrics: area under the receiver operator characteristic curve (AU ROC) and true positive rate at a true negative rate of 95% (TPR (TNR 95%)).
- In both metrics, higher scores are better.



Detecting Out-of-Distribution Inputs



- Our model safeguard achieves high AU ROC and TPR (TNR 95%) scores for each novel attack and the concept drift scenario.
- We note slightly worse results on UNSW-NB15.
 - The discriminative classifier did slightly better on predicting benign samples despite the distribution shift.
 - Thus, there may have been more overlap between the in-distribution and out-of-distribution samples.

Layer	AU ROC ($\uparrow$)	TPR (TNR 95%) ($\uparrow$)
FTP-Patator		
linear1	0.9957 (± 0.0008)	100.00% ($\pm 0.00\%$)
linear2	0.9792 (± 0.0060)	95.84% ($\pm 7.63\%$)
linear3	0.9644 (± 0.0137)	77.51% ($\pm 19.94\%$)
Infiltration		
linear1	0.9614 (± 0.0132)	70.60% ($\pm 21.01\%$)
linear2	0.9742 (± 0.0149)	86.69% ($\pm 15.44\%$)
linear3	0.9528 (± 0.0358)	66.26% ($\pm 28.85\%$)
SSH-Patator		
linear1	0.9921 (± 0.0014)	100.00% ($\pm 0.00\%$)
linear2	0.9742 (± 0.0055)	94.58% ($\pm 7.02\%$)
linear3	0.9513 (± 0.0125)	64.12% ($\pm 15.48\%$)
UNSW-NB15		
linear1	0.9263 (± 0.0071)	77.83% ($\pm 0.39\%$)
linear2	0.9536 (± 0.0072)	77.96% ($\pm 3.03\%$)
linear3	0.9583 (± 0.0090)	77.71% ($\pm 4.49\%$)

Accounting for Real-World Data Imbalance



- In real-world scenarios, the amount of benign network traffic far exceeds the amount of malicious network traffic.
- Our previous results contain an equal number of benign and malicious packets during inference.
- With a heavy class imbalance of 46.8 benign packets to every 1 malicious one, our discriminative classifier's accuracy increases to 99.63%.

Accounting for Real-World Data Imbalance

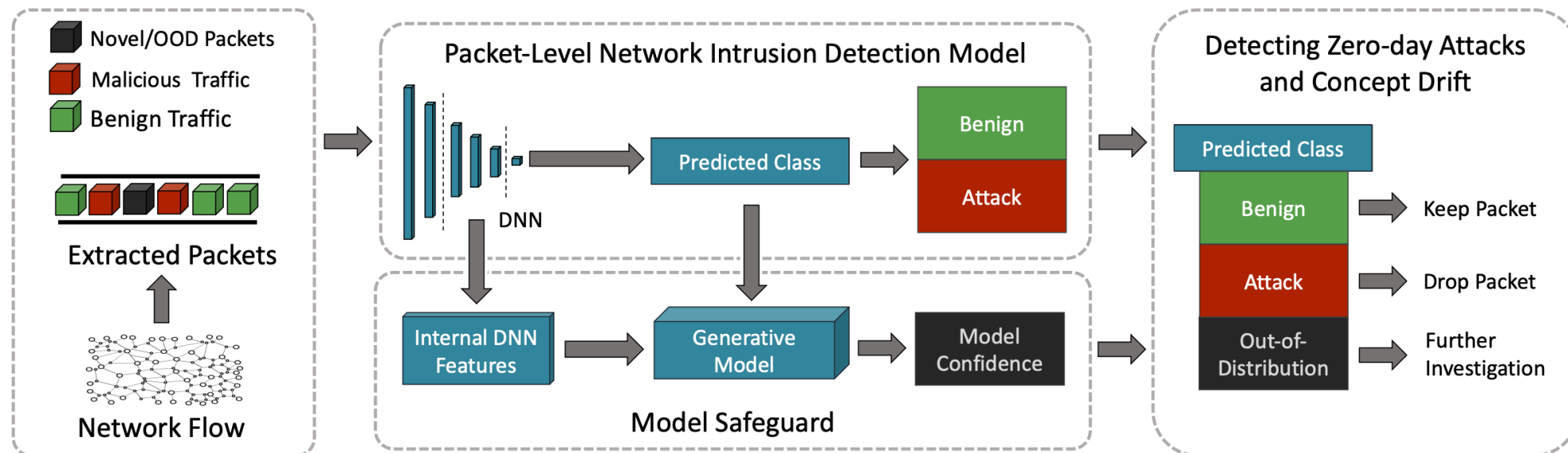


- For the model safeguard evaluation, we have ratios of in-distribution to out-of-distribution samples at 1623.9:1, 1953.5:1, and 645.8:1 for FTP-Patator, Infiltration, and SSH-Patator, respectively.
- We still achieve high AU ROC scores, albeit with slight (expectedly so) performance degradation.

OOD Attack	Layer	AU ROC ($\uparrow$)
FTP-Patator	linear1	0.9936 (± 0.0014)
Infiltration	linear2	0.9668 (± 0.0182)
SSH-Patator	linear1	0.9901 (± 0.0007)

Conclusions

- Transformer architectures can accurately predict benign packets from malicious ones when presented with payloads **from** the training distribution.
- These models **fail** to generalize to OOD samples such as zero-day attacks or shifting distributions.
- A model safeguard can model the distributions of internal features of the DNNs for in-distribution data to provide a **confidence score** that operators can trust the DNN.



Computational Overhead



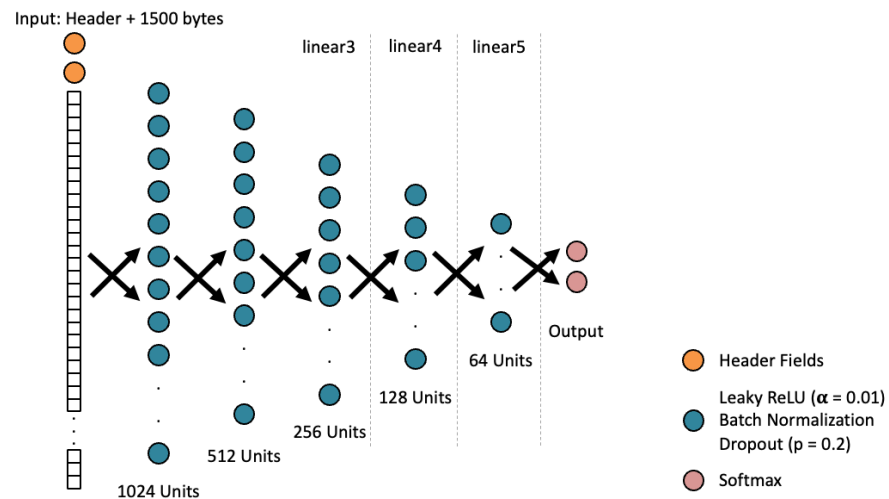
- Our current transformer architecture adds significant computational overhead with an average throughput ~ 415 packets per second per GPU.
 - Future designs could reduce the number of transformer blocks to one or reduce the embedding size.
- The normalizing flows model safeguard has comparatively high throughput, processing $\sim 24,000 - 31,000$ packets per second per GPU, depending on the dimensionality of the extracted layer.

Practical Considerations

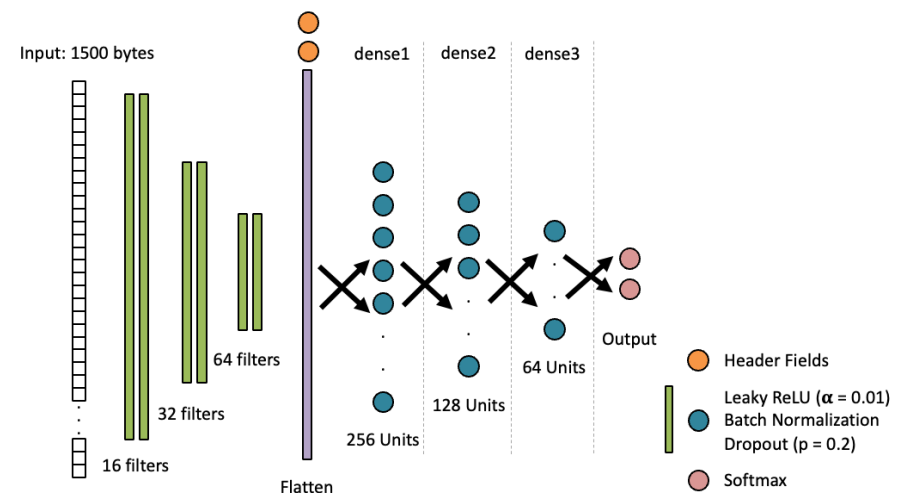


- To reduce the overhead, we can use simpler ML models for the discriminative classifier.
- Similarly, we can use more traditional out-of-distribution detection methods instead of normalizing flows.
- These other models have slightly worse results but with significantly less latency.

Fully-Connected Neural Network



Convolutional Neural Network



Practical Considerations



- One could design the network intrusion detection system in such a way that connections to more critical devices use the more accurate but slower models whereas less critical, high throughput traffic use the less accurate but faster models.

Architecture	Latency (↓)
CNN	172.48 μ sec
FNN	144.60 μ sec
Transformer	2418.02 μ sec

Gaussian Kernel Density		Normalizing Flows	
Feature Dimensionality	Latency (↓)	Feature Dimensionality	Latency (↓)
256	59.60 μ sec	256	40.45 μ sec
128	29.59 μ sec	128	35.37 μ sec
64	23.96 μ sec	64	32.95 μ sec