Net-GPT: A LLM-Empowered Man-in-the-Middle Chatbot for Unmanned Aerial Vehicle

Brett Piggott[1], Siddhant Patil[2], Guohuan Feng[1], Ibrahim Odat[1], Rajdeep Mukherjee[1], Balakrishnan Dharmalingam[1], Anyi Liu[1]
[1]Department of Computer Science and Engineering, Oakland University, Rochester, Michigan, USA
[2]Department of Computer Science, University of Wisconsin, Madison, Wisconsin, USA

ABSTRACT

In the dynamic realm of AI, integrating Large Language Models (LLMs) with security systems reshape cybersecurity. LLMs bolster defense against cyber threats but also introduce risks, aiding adversaries in generating malicious content, discovering vulnerabilities, and distorting perceptions. This paper presents Net-GPT, an LLM-empowered offensive chatbot that understands network protocols and launches Unmanned Aerial Vehicles (UAV)-based Man-in-the-middle (MITM) attacks against a hijack communication between UAV and Ground Control Stations (GCS). Facilitated by an edge server equipped with finely tuned LLMs, Net-GPT crafts mimicked network packets between UAV and GCS. Leveraging the adaptability of popular LLMs, Net-GPT produces context-aligned network packets. We fine-tune and assess Net-GPT's LLM-based efficacy, showing its impressive generative accuracy: 95.3% for Llama-2-13B and 94.1% for Llama-2-7B. Smaller LLMs, such as Distil-GPT-2, reach 77.9% predictive capability of Llama-2-7B but are 47× faster. Cost-efficiency tests highlight model quality's impact on accuracy while fine-tuning data quantity enhances predictability on specific metrics. It holds great potential to be used in edge-computing environments with amplified computing capability.

CCS CONCEPTS

• **Networks → Protocol testing and verification**; • **Computing methodologies → Natural language generation**; • **Security and privacy → Security protocols**.

KEYWORDS

Man-in-the-Middle (MITM), Large Language Model, System Security, and Cyber Attack

1 INTRODUCTION

In the evolving landscape of artificial intelligence, merging LLMs with system and software security has ushered in an unstoppable transformation. On the one hand, with their superior capabilities to comprehend and generate language at unprecedented scales, LLMs hold the promise of amplifying detective and defensive capabilities against cyber adversaries [2, 5, 8, 16, 17, 20]. On the other hand, LLMs can also be misused by adversaries to produce malicious payload [1], inject faults [6], construct backdoors [19], and manipulate people's minds [2].

As a standard offensive procedure, adversaries first gain knowledge of system and hardware design and software vulnerabilities. Then, they launch offensive tools to exploit the discovered vulnerabilities. Finally, they compromise the system and gain access. However, with the aid of LLMs, the capability of adversaries can be significantly escalated. For example, the malicious autonomous agent learns from vast data, discerns patterns, and generates exploits automatically. Furthermore, leveraging LLMs significantly amplifies the adversary's ability to understand the semantics and context of a situation and identifies a software's functionalities and their relations. Thus, reducing the window of launching exploits makes intrusion detection and response more difficult.

In this paper, we present Net-GPT, a malicious chatbot designed to understand network protocols and perform man-in-the-middle attacks on UAVs. Net-GPT allows the malicious UAV to intercept and control communications between a regular UAV and its GCS. By doing so, it can impersonate the benign UAV, exchanging information with the GCS and vice versa. Net-GPT derives its ability to create mimicked network packets from an edge server that uses fine-tuned LLMs. Training data from a public network traffic repository is employed to enhance LLMs' capabilities. While amplifying this dataset using intercepted traffic between the regular UAV and GCS is feasible, the research indicates this is optional. Due to the high generative capacities of LLMs, they can craft appropriate network packets in line with the existing communication context.

Our experiments show promising results of using LLMs as an effective tool for the adversary to generate and sustain exploits. We demonstrate that LLMs, including Llama-2-13B and Llama-2-7B, reach 95.3% and 94.1%, respectively, prediction accuracy like an actual UAV or GCS. After fine-tuning, smaller LLMs, such as Distil-GPT-2, can achieve 77.9% predictive capability of Llama-2-7B and 76.7% of Llama-2-13B, respectively. The quantity of fine-tuning datasets is essential: it may play a minor role in the overall fine-tuning results. But, it can play a significant role in improving the predictive accuracy of some values.

The contributions of this paper are summarized as follows:

- We designed and implemented Net-GPT that understands network protocols and performs man-in-the-middle attacks through UAVs.
- We designed and implemented attacks that hijack the benign UAV and GCS communication session.
- We evaluated the effectiveness of fine-tuning LLMs used by Net-GPT to mimic the conversation between the UAV and GCS with the support of open-source LLMs.

SEC'23, December 6-9, 2023, Wilmington, DE
© 2018 Association for Computing Machinery.
ACM ISBN 979-8-4007-0123-8/23/12…$15.00
https://doi.org/10.1145/3583740.3626809

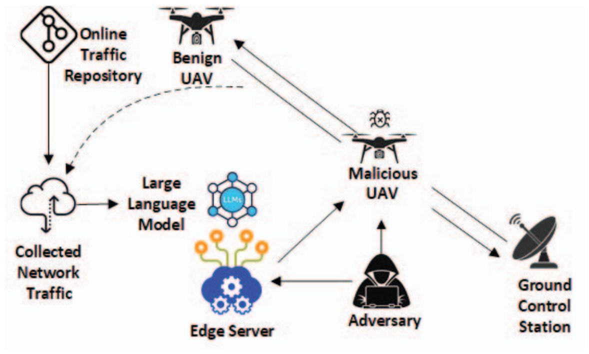


Figure 1: The working mechanism of Net-GPT.

We organize this paper as follows: Section 2 researches the related work in the field. Section 3 briefly describes the capabilities of an adversary and the attacking scenario. Section 4 provides the detailed steps of constructing Net-GPT. Section 5 presents the experimental results. Section 6 concludes the paper and suggests our future directions.

2 RELATED WORK

This section briefly reviews the related work that integrates LLMs with system and software security, categorizing *defensive* and *offensive* sides.

On the *defensive* side, Wang et al. [20] explores the strengths and weaknesses of using ChatGPT [15] in cyber security. They suggested valuable applications on ChatGPT for Software Security. Crothers et al. provide an extensive survey on threat models posed by LLMs and machine-generated text detection methods with cyber security and social context[2]. Pearce et al.[16] uses LLMs in reverse engineering software without comments in the decompiled code. They leverage OpenAI Codex [14] to understand the structure and functionalities of software. Jain et al.[8] presents the capability of LLMs in generating code from natural language specification. Their tool, namely *jigsaw*, synthesizes Python code with LLMs and multi-modal specifications. The limitation of this work is that the quality of the generated code might not be high if the modal specifications are ambiguous. Ferrag et al. [5] introduces *SecurityLLM*, a pre-trained LLM for cyber-threat detection. Although it detects threats and responds effectively, it needs constant fine-tuning with updated datasets to detect real-world exploits. Sandoval et al. [17] analyze the code generated with LLM for vulnerability detection. The number of the identified vulnerabilities is low for a simple application, while the number could be high for a complex application.

On the *offensive* side, Sai Charan et al. explore the misuse of the ChatGPT for attacking payload generation[1]. ChatGPT generates code that creates malicious payloads for the top 10 MITRE techniques prevalent in 2022 and emphasizes mitigating the risks with LLMs. Fortmann uses LLM to generate code for the hardware to inject faults[6]. This system also relies on the ChatGPT and the fine-tuned model to generate injected fault injection to build a more robust system. Shi et al. [19] propose a backdoor, namely *BadGPT*, that uses reinforcement learning (RL) to generate manipulated text on movie reviews.

3 THREAT MODEL

We assume that the adversary targets two goals: 1) hijacking the benign UAV and 2) playing the MITM attack and controlling it. For the first goal, the malicious UAV has joined the same network as the benign UAV and GCS before launching the attack. The benign UAV and GCS communicate with the TCP protocol on the network layer. Although the malicious UAV might not have sufficient computing capability to run LLMs, it can leverage the computing resources (e.g., the GPU) from a nearby edge server. Since the inquiry sent from the malicious UAV and the edge server is generic and benign, perimeter defensive measures, including authentication, firewall, and intrusion detection system, might not be able to tell the intent of the malicious UAV. Therefore, the edge server must not be compromised or work as a colluder. To fulfill the second goal, the malicious UAV impersonates the GCS towards benign UAV by sending crafted packets, and vice versa. Further, to ensure network packets are crafted within a sufficiently long time window, the malicious UAV can send signaling packets, such as delayed acknowledgment, duplicate acknowledgment (DupACK), and Keep-Alives, to keep the connection alive and gain more time in the game.

4 SYSTEM DESIGN

To successfully launch an MITM attack, we take two steps: First, we launch an attack to hijack a benign UAV. Then, we let the malicious UAV connect to an LLM-empowered edge server to obtain mimicked data. We elaborate on their working mechanism in the following subsections. The attacking scenario is illustrated in Fig. 1

4.1 Hijacking Benign UAV

To hijack the benign UAV, we developed three attacking scenarios, namely *ARP poisoning*, *semi-session-hijacking* and *full session-hijacking*, as illustrated in the Fig. 2. The adversary uses various tools to launch exploits. For example, it sends de-authenticate packets to GCS, causes the GCS to disconnect, and then sends malicious MAVLink control messages to PX4. Table 1 shows the launched attacks affect different security criteria, including confidentiality (C), privacy (P), integrity (I), and availability (A). We encourage interested readers to refer to our publication [4] of these attacks for details.

4.2 Fine-tuning A Model

Before hijacking the benign UAV, the adversary should obtain sufficient network traffic to fine-tune general-purposed LLMs. To make the fine-tuning data representative enough to avoid bias or overfitting, we diversify the dataset and keep its size at a reasonable scale. In our evaluation, we collected 6971 TCP sessions with a total number of 100K packets.

For the fine-tuning data, we compose the fine-tuning data in the format as illustrated in Fig. 3. Each data entry includes three sections: `#Previous_Packet`, `#Predicted_Packet`, and `#Context`, which use the same `#BLOCK` section that specifies `<key:value>` pairs.

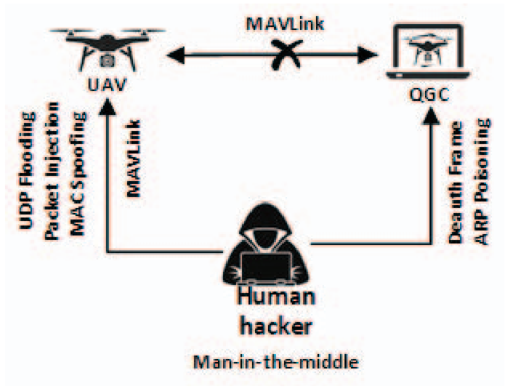
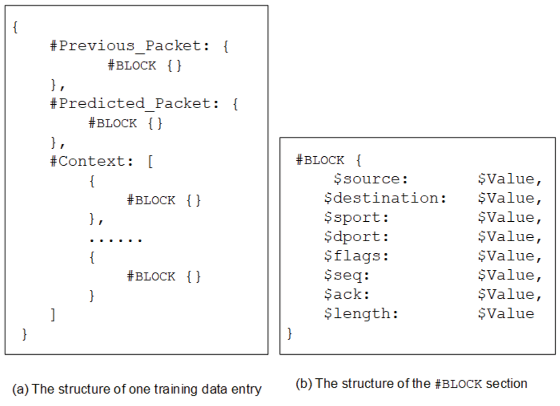


Figure 2: Cyber attacks launched against the UAV.

Table 1: Attacks launched against PX4 autopilot.

Attack Method	Unauth Access	Man-in-the-middle	Denial of Service	Affected Criteria
MAC Spoofing	✓	✓	×	C, P, I
Session Hijacking	✓	✓	×	C, P, I
ARP Poisoning	✓	✓	×	C, P, I
Packet Injection	✓	✓	×	C, P, I
Flooding Attack	×	×	✓	A
De-authentication	×	✓	✓	A



```
{
    #Previous_Packet: {
        #BLOCK {}
    },
    #Predicted_Packet: {
        #BLOCK {}
    },
    #Context: [
        {
            #BLOCK {}
        },
        ......
        {
            #BLOCK {}
        }
    ]
}
```

(a) The structure of one training data entry

```
#BLOCK {
    $source:         $Value,
   $destination:     $Value,
   $sport:           $Value,
   $dport:           $Value,
   $flags:           $Value,
   $seq:             $Value,
   $ack:             $Value,
   $length:          $Value
}
```

(b) The structure of the #BLOCK section

Figure 3: The Format of Data Under the fine-tuning.

As we choose four LLMs as the base model, the fine-tuning process uses Quantized Low Rank Adapters (QLORA) [3], an alternative approach of Low Rank Adapters (LoRA) [7], as our fine-tuner. The parameters used by the QLORA fine-tuner are listed in Table 2.

Table 2: Parameters used by QLORA fine-tuner.

Parameter	Value	Parameter	Value
NF4 quantization	nf4	**Logging steps**	10
Batch size/device	8	**Learning rate**	2e-4
Gradient steps	12	**Global grad norm**	0.3
Paged Optimizer	paged_adamw_32bit	**Warm-up ratio**	0.03
Gradient steps	12	**Scheduler type**	constant
Gradient steps	12	**Epoch(s)**	1 - 10

5 EVALUATION

We use the network traffic data collected from a public repository [9], from which *bigFlows* is used for fine-tuning and *smallFlows* for testing. The statistics of the dataset are listed in Table 3.

Table 3: The statistics of dataset.

	fine-tuning Data (*bigFlows*)	Testing Data (*smallFlows*)
Size	368 MB	9.4 MB
# of Packets	791615	14261
# of Flows	40686	1209
# of Applications	132	28
Average packet size	449 bytes	646 bytes
Duration	5 minutes	5 minutes

Table 4 lists the base models we fine-tuned. We choose these models to try to answer at least three research questions (RQs): RQ_1) *Does the larger number of parameters of an LLM significantly improve the effectiveness?* RQ_2) *What pivot point balances the quantity of dataset and the number of training epochs?* RQ_3) *Would a smaller LLM produce similar effectiveness as a larger LLM regarding the number of parameters?*

Table 4: The LLMs used in the Evaluation.

Model Name	sub-version	# of paramters
Llama-2-7B	*llama-2-7b-hf* [11]	7 billion
Llama-2-13B	*llama-2-13b-hf* [10]	13 billion
GPT-2	*GPT-2* [13]	137 million
Distil-GPT-2	*distilgpt2* [12]	82 million

We use `scapy` [18] to generate network packets based on the responses returned from the edge server. The edge server is equipped with ASUS GeForce RTX 4090 as the computing unit. We extract the TCP session by using `tshark` [21]. We wrote 350 lines of Python code to pre-process the data and generate .JSON file for fine-tuning.

Table 5: The effectiveness of generating response (Llama-2-13B, 1 epoch) - In Percentage.

Model Name	Src_IP	Dst_IP	Src_Port	Dst_Port	Flag	Seq#	Ack#	Length	Overall Average
Llama-2-13B (20k)	98.851	98.851	98.851	98.851	88.506	96.552	87.356	100	95.977
Llama-2-13B (40k)	97.590	97.594	97.590	97.590	85.542	96.386	87.952	98.795	94.88
Llama-2-13B (60k)	98.824	98.824	98.824	98.824	84.706	97.647	95.294	98.824	96.471
Llama-2-13B (80k)	97.675	97.675	97.675	97.675	77.612	92.537	91.045	98.508	93.470
Llama-2-13B (100k)	98.851	98.851	98.851	98.851	80.46	97.701	93.103	98.851	95.69
Mean	98.226	98.226	98.226	98.226	83.365	96.165	90.950	98.995	95.297
Standard Deviation	0.867	0.867	0.867	0.867	4.316	2.116	3.37	0.578	1.173

Table 6: The effectiveness of generating response (Llama-2-7B, 1 epoch) - In Percentage.

Model Name	Src_IP	Dst_IP	Src_Port	Dst_Port	Flag	Seq#	Ack#	Length	Overall Average
Llama-2-7B (20k)	92.5	95	93.75	95	83.75	92.5	80	93.75	90.781
Llama-2-7B (40k)	100	100	100	100	85.542	95.181	90.361	98.795	96.235
Llama-2-7B (60k)	100	100	100	100	86.25	98.75	93.75	97.5	97.031
Llama-2-7B (80k)	96.296	97.531	96.296	97.531	80.247	93.827	88.889	95.062	93.2099
Llama-2-7B (100k)	97.531	97.531	97.531	97.531	74.074	96.296	86.42	98.765	93.21
Mean	97.265	98.012	97.515	98.012	81.973	95.311	87.884	96.775	94.094
Standard Deviation	3.11	2.088	2.646	2.088	4.989	2.394	5.144	2.272	2.537

5.1 Effectiveness

To answer the RQ_1, we first analyzed the correctness that Net-GPT generates counterfeit network packets. Table 5 illustrates the correction rate (in percentage) for Net-GPT to predict TCP packet fields, given different quantities of data (ranging from 20k to 100k) for Llama-2-13B with one epoch of fine-tuning. From this experiment we have the following two observations. First, the rates that Net GPT predicts counterfeit network packet fields are high, ranging between 98.995 and 83.365. In addition, the low standard deviation of field prediction indicates that outliers of incorrect predictions are very low. Second, the datasets' quantities contribute little to the prediction accuracy. Similarly, we observe similar results when we use Llama-2-7B, as illustrated in Table 6. It is impressive that for specific test cases, for instance, with the dataset size of 40K and 60k the correctness of prediction is 100%, which is higher than those of Llama-2-13B. However, the predictions of Llama-2-7B also show higher standard deviations, which indicate its weaker stability.

Then, we analyze the generative error generated by Net-GPT Table 7 shows various errors for fine-tuning Llama-2-13B, whose averages range between 20.307% and 0%. From this experiment, we also make the following two observations. First, we observe that most errors are just attributed to one field, which the adversary can quickly fix. Second, the field that produces the highest error rate is the *flag* field. The reason for the error is two-fold: 1) there are some *out-of-the-order* TCP packets in a session, which is used as the context for fine-tuning. They need to be re-ordered, and 2) some packets in a session are mistakenly used as the context of response generation. These packets, which have no data payload or are primarily for controlling or signaling purposes, should be excluded from the fine-tuning dataset or grouped into a particular context to suppress error rates. We will address this issue in our future research.

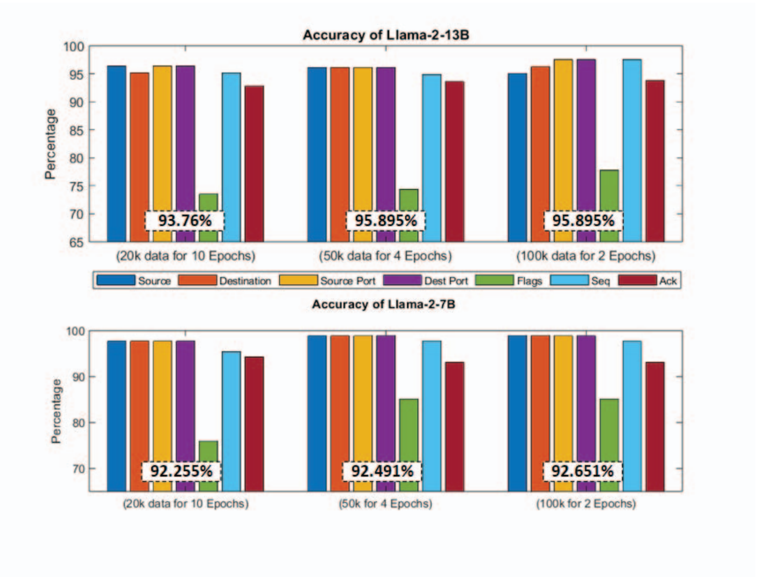


Figure 4: The cost-efficiency test for three testing combinations (Llama-2-13B and Llama-2-7B)

Table 7: The analysis of generative errors (Llama-2-13B, 1 epoch) - In Percentage.

Model Name	0 err.	1 err.	2 errs.	3 - 5 errs.	6 errs.	7 errs.	8 errs.	Flags	Seq#	Ack#
Llama-2-13B (20k)	77.012	18.391	3.448	0	1.149	0	0	8.046	2.299	8.046
Llama-2-13B (40k)	75.904	19.277	2.41	0	0	2.41	0	9.639	0	9.639
Llama-2-13B (60k)	80	17.647	1.177	0	0	1.177	0	12.941	1.177	3.529
Llama-2-13B (80k)	74.419	20.930	2.326	0	0	1.163	1.163	13.954	1.163	5.814
Llama-2-13B (100k)	73.563	25.287	0	0	0	0	1.149	18.391	1.149	5.747
Mean	76.179	20.307	1.872	0	0.23	0.95	0.462	12.594	1.158	6.555
Standard Deviation	2.516	3.041	1.32	0	0.514	1.004	0.633	4.031	0.813	2.35

Table 8: The effectiveness of generative errors (Llama-2-7B, 1 epoch) - In Percentage.

Model Name	0 err.	1 err.	2 errs.	3 errs.	4 errs.	5 errs.	6 errs.	7 errs.	8 errs.	Flags	Seq#	Ack#
Llama-2-7B (20k)	68.750	17.5	5	2.5	0	0	5	1.25	0	6.25	1.25	8.75
Llama-2-7B (40k)	73.494	24.096	1.205	1.205	0	0	0	0	0	12.048	4.819	7.228
Llama-2-7B (60k)	78.75	18.75	2.5	0	0	0	0	0	0	11.25	1.25	6.25
Llama-2-7B (80k)	69.136	23.457	2.469	1.235	1.235	0	1.235	0	1.235	0	0	0
Llama-2-7B (100k)	64.198	29.63	3.704	0	0	0	1.235	0	1.235	20.988	0	8.642
Mean	70.866	22.687	2.9755	0.988	0.247	0	1.494	0.25	0.494	10.107	1.464	6.174
Standard Deviation	5.5	4.827	1.436	1.042	0.552	0	2.055	0.559	0.676	7.754	1.977	3.604

Table 9: The analysis of generative errors (GPT-2) - In Percentage.

Model Name	0 err.	1 err.	2 errs.	3 errs.	4 errs.	5 errs.	6 errs.	7 errs.	8 errs.	Flags	Seq#	Ack#
GPT-2 (20k, 1 Epoch)	33.333	20.69	14.943	8.046	4.598	4.598	12.644	1.149	0	2.299	18.391	0
GPT-2 (50k, 1 Epoch)	42.529	14.943	16.092	5.747	4.598	5.747	5.747	3.448	1.149	1.149	4.598	8.046
GPT-2 (100k, 1 Epoch)	47.126	10.345	18.391	5.747	4.598	0	6.897	3.448	3.448	0	0	5.747
GPT-2 (100k, 5 Epoch)	26.744	22.093	13.954	4.651	1.163	1.163	19.767	5.814	4.651	1.163	4.651	5.814
GPT-2 (100k, 10 Epoch)	11.494	20.69	12.644	9.195	6.897	5.747	25.287	5.747	2.299	1.149	4.598	4.598
Mean	32.245	17.752	15.205	6.677	4.371	3.451	14.068	3.921	2.31	1.15	6.448	4.841
Standard Deviation	14.047	4.972	2.187	1.874	2.051	2.693	8.382	1.94	1.834	0.813	6.969	2.981

Table 10: The effectiveness of generating response (GPT-2) - In percentage.

Model Name	Src_IP	Dst_IP	Src_Port	Dst_Port	Flag	Seq#	Ack#	Length	Overall Avg.
GPT-2 (20k, 1 Epoch)	74.713	75.862	78.161	78.161	78.161	44.828	73.563	96.552	75
GPT-2 (50k, 1 Epoch)	58.621	81.609	81.609	81.609	80.46	71.264	71.264	94.253	77.586
GPT-2 (100k, 1 Epoch)	56.322	82.759	83.908	83.908	82.759	78.161	63.218	93.103	78.017
GPT-2 (100k, 5 Epoch)	41.861	65.116	67.442	67.442	83.721	60.465	48.837	94.186	66.134
GPT-2 (100k, 10 Epoch)	34.483	50.575	63.218	63.218	70.115	47.126	41.379	89.655	57.471
Mean	53.2	71.184	74.8687	74.868	79.043	60.369	59.653	93.55	70.842
Standard deviation	15.659	13.473	9.067	9.067	5.436	14.6	14.071	2.514	8.877
Capability of Llama-2-13B	54.088	72.373	76.118	76.118	93.963	62.562	65.471	94.436	74.158
Capability of Llama-2-7B	54.695	72.628	76.775	76.386	96.426	63.339	67.876	96.668	75.289

5.2 Cost-efficiency between Data Size and Epochs

A lesson we learned from the previous sets of experiments is that the fine-tuning process is time-consuming. Given the same fine-tuning parameters and hardware setting, we observe that two factors are crucial. That is the quantity of dataset and the number of fine-tuning epochs. Thus, our second experiment evaluates the effectiveness and finds a pivot point for these two factors (the answer of RQ_2). For each model, we construct three parameter combinations (20K data and ten epochs; 50K data and four epochs; 100K data and two epochs). Fig. 4 illustrates their predictive accuracy. As expected, the higher-parameter model (Llama-2-13B) outperforms

Table 11: The analysis of generative errors (Distil-GPT-2) - In Percentage.

Model Name	0 err.	1 err.	2 errs.	3 errs.	4 errs.	5 errs.	6 errs.	7 errs.	8 errs.	Flags	Seq#	Ack#
Distil-GPT-2 (20k, 1 Epoch)	32.184	18.391	18.391	8.046	13.793	3.448	4.598	1.149	0	0	14.943	0
Distil-GPT-2 (50k, 1 Epoch)	28.736	13.793	19.54	13.793	11.494	4.598	5.747	2.299	0	0	8.046	0
Distil-GPT-2 (100k, 1 Epoch)	26.437	5.747	21.839	8.046	8.046	16.092	10.345	3.448	0	0	0	0
Distil-GPT-2 (100k, 5 Epoch)	36.145	13.253	21.687	7.229	4.819	2.41	4.819	8.434	0.012	0	0	1.205
Distil-GPT-2 (100k, 10 Epoch)	32.184	18.391	18.391	8.046	13.793	3.448	4.598	1.149	0	0	14.943	0
Mean	31.137	13.915	19.97	9.032	10.389	5.999	6.021	3.296	0.002	0	7.586	0.241
Standard Deviation	3.712	5.178	1.704	2.685	3.902	5.695	2.463	3.026	0.005	0	7.476	0.539

Table 12: The effectiveness of generating response (Distil-GPT-2) - In Percentage.

Model Name	Src_IP	Dst_IP	Src_Port	Dst_Port	Flag	Seq#	Ack#	Length	Overall Avg.
Distil-GPT-2 (20k, 1 Epoch)	85.058	81.609	63.218	74.713	82.759	47.126	82.759	95.402	76.58
Distil-GPT-2 (50k, 1 Epoch)	83.908	48.276	77.012	81.609	83.908	37.931	78.161	95.402	73.276
Distil-GPT-2 (100k, 1 Epoch)	59.770	41.379	67.816	67.816	75.862	43.678	72.414	98.851	65.948
Distil-GPT-2 (100k, 5 Epoch)	51.807	69.880	83.133	83.133	78.313	71.084	62.651	92.771	74.096
Distil-GPT-2 (100k, 10 Epoch)	85.058	81.609	63.218	74.713	82.759	47.126	82.759	95.402	76.58
Mean	73.12	64.551	70.879	76.397	80.72	49.389	75.749	95.566	73.296
Standard Deviation	16.077	18.789	8.867	6.161	3.459	12.697	8.467	2.161	4.365
Capability of Llama-2-13B	74.341	65.629	72.062	77.672	95.956	51.183	83.137	96.472	76.727
Capability of Llama-2-7B	75.176	65.86	72.685	77.946	98.472	51.819	86.192	98.751	77.897

the smaller-parameter model (Llama-2-7B) among three parameter combinations. It is clear that for both Llama-2-13B and Llama-2-7B, the overall predictive accuracy is the best for the testing combination with the largest dataset and the smallest number of epochs. For example, combining 100K data and two epochs generates the highest accuracy. Even though the accuracy for the three combinations is very close, it is interesting that the larger dataset remarkably improves the predictive accuracy of the "*Flags*" field, which plays an essential role in sustaining the communication session. Therefore, using a larger dataset to fine-tune a model to keep the network communication stable is still worthwhile.

5.3 Using Smaller LLMs for the Prediction

Our first two experiments use LlaMa-2 models (7B and 13B) deployed on the edge server. To answer RQ_3 is particularly important for at least three reasons. First, it will allow us to deploy smaller LLMs on mobile and portable devices, which are limited in storage and computing capability. Second, smaller LLMs need less time to fine-tune and are much more responsive to generate the prediction, which is much more favorable for real-world applications. Finally, if a computing task can be divided into multiple sub-tasks, in which some sub-tasks are executed on board with smaller LLMs while others are offloaded to the edge equipped with larger LLMs, the overall performance will be improved. Table 9 and Table 10 demonstrate the predictive capability of GPT-2 is approximately 74.158% of Llama-2-13B, while 75.289% of Llama-2-7B. In addition, the average response time for GPT-2 is 5.18 seconds, compared to 108.311 seconds for Llama-2-13B, which is 21× faster. Our experiments on Distil-GPT-2, which has fewer parameters than GPT-2, demonstrate better results as illustrated in Table 11 and Table 12. It reaches approximately 76.727% of Llama-2-13B's predictive capability, while 77.897% of Llama-2-7B. Moreover, the average response time for Distil-GPT-2 is 2.3136 and 47× faster than Llama-2-13B.

These results show the promise that smaller LLMs can take a moderate amount of jobs with less rigorous accuracy requirements but a quick response time. Finally, it is interesting that the increasing number of epochs does not improve the accuracy, but just the opposite. The results imply that smaller LLMs cannot improve their predictive accuracy with a longer training time.

6 CONCLUSION

The convergence of Large Language Models (LLMs) with security systems transforms cybersecurity in the AI landscape. This paper introduces Net-GPT, an offensive chatbot that understands network protocol and exploits UAV with MITM attacks. Net-GPT empowers offensive UAVs to intercept and control benign UAV-GCS communications. Powered by finely-tuned LLMs, Net-GPT generates mimicked network packets in line with benign communication. Fine-tuning showcases Net-GPT's efficacy and model quality's influence on accurate prediction, while its potential resides in augmented edge computing.

REFERENCES

[1] P. V. Sai Charan, Hrushikesh Chunduri, P. Mohan Anand, and Sandeep K Shukla. 2023. From Text to MITRE Techniques: Exploring the Malicious Use of Large Language Models for Generating Cyber Attack Payloads. arXiv:2305.15336

[2] Evan N. Crothers, Nathalie Japkowicz, and Herna L. Viktor. 2023. Machine-Generated Text: A Comprehensive Survey of Threat Models and Detection Methods. *IEEE Access* 11 (2023). https://doi.org/10.1109/ACCESS.2023.3294090

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314

[4] Balakrishnan Dharmalingam, Ibrahim Odat, Rajdeep Mukherjee, Brett Piggott, and Anyi Liu. 2023. Heterogeneous Generative Dataset for UASes. In *Proceedings of the First IEEE Conference on Mobility: Operations, Services, and Technologies (IEEE MOST'23)*. IEEE, New York, NY, USA. https://doi.org/10.1109/MOST57249.2023.00034

[5] Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, Lucas C. Cordeiro, Merouane Debbah, and Thierry Lestable. 2023. Revolutionizing Cyber Threat Detection with Large Language Models. arXiv:2306.14263

[6] Sara Fortmann. 2023. Leveraging ChatGPT to build a Hardware Fault Injection Tool. https://onekey.com/blog/leveraging-chatgpt-to-build-a-hardware-fault-injection-tool

[7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *CoRR* abs/2106.09685 (2021). arXiv:2106.09685

[8] Naman Jain, Skanda Vaidyanath, Arun Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram Rajamani, and Rahul Sharma. 2022. Jigsaw: Large Language Models Meet Program Synthesis. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 1219–1231. https://doi.org/10.1145/3510003.3510203

[9] Fred Klassen and AppNeta. Accessed: August 16, 2023. Tcpreplay Network Traffic Repository. https://tcpreplay.appneta.com/wiki/captures.html.

[10] Meta. Accessed: August 19, 2023. Llama 2 13B. https://huggingface.co/meta-llama/Llama-2-13b-hf/.

[11] Meta. Accessed: August 19, 2023. Llama 2 7B. https://huggingface.co/meta-llama/Llama-2-7b-hf.

[12] OpenAI. Accessed: August 19, 2023. distilgpt2. https://huggingface.co/distilgpt2.

[13] OpenAI. Accessed: August 19, 2023. GPT 2. https://huggingface.co/gpt2.

[14] OpenAI. Accessed: August 19, 2023. OpenAI Codex. https://openai.com/blog/openai-codex.

[15] OpenAI. Accessed: August 20, 2023. ChatGPT - Chat Generative Pre-trained Transformer. https://chat.openai.com/.

[16] Hammond Pearce, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Brendan Dolan-Gavitt. 2022. Pop Quiz! Can a Large Language Model Help With Reverse Engineering? *CoRR* abs/2202.01142 (2022). arXiv:2202.01142

[17] Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. 2023. Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants. arXiv:2208.09727

[18] Scapy Community. Accessed: August 16, 2023. Scapy: A powerful interactive packet manipulation libary. https://scapy.net/.

[19] Jiawen Shi, Yixin Liu, Pan Zhou, and Lichao Sun. 2023. BadGPT: Exploring Security Vulnerabilities of ChatGPT via Backdoor Attacks to InstructGPT. arXiv:2304.12298

[20] Zhilong Wang, Lan Zhang, and Peng Liu. 2023. ChatGPT for Software Security: Exploring the Strengths and Limitations of ChatGPT in the Security Applications. arXiv:2307.12488

[21] Wireshark. Accessed: August 19, 2023. tshark. https://www.wireshark.org/docs/man-pages/tshark.html.