

# Predict, Prioritize, Patch

How Microsoft Harnesses LLMs for Security Response

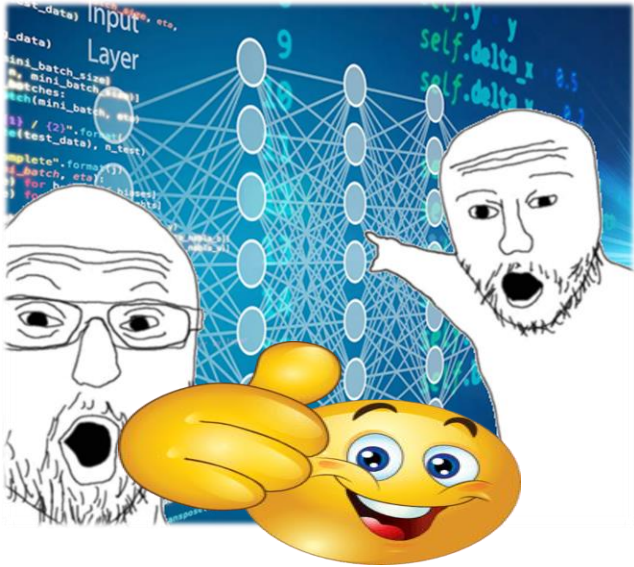
Bill Demirkapi

# Who Am I?



- Lead Emerging Threats at the Microsoft Security Response Center.
- Cloud and low-level software security background.
- Experimented with Large Language Models for a little over a year.
- I am not a machine learning expert.

# Agenda

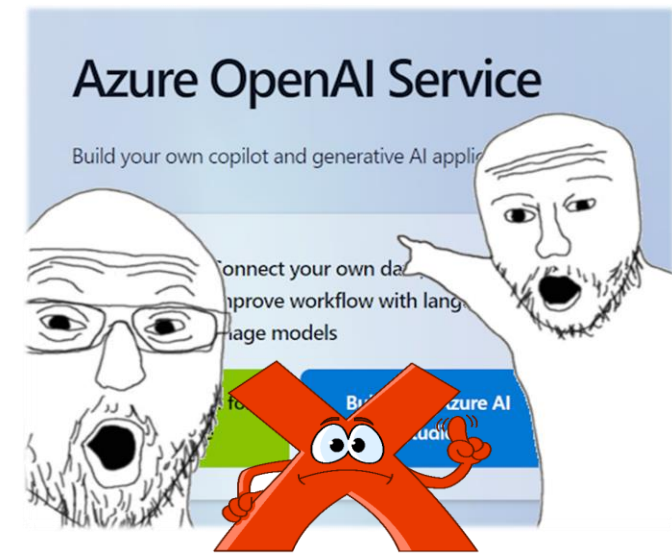


## What Is This Talk About?

- What are Large Language Models and their advantages?
- Why is MSRC interested in them?
- How can you apply them for security response?
- What are they capable of? What's our take on their future?
- How do you get the right data?

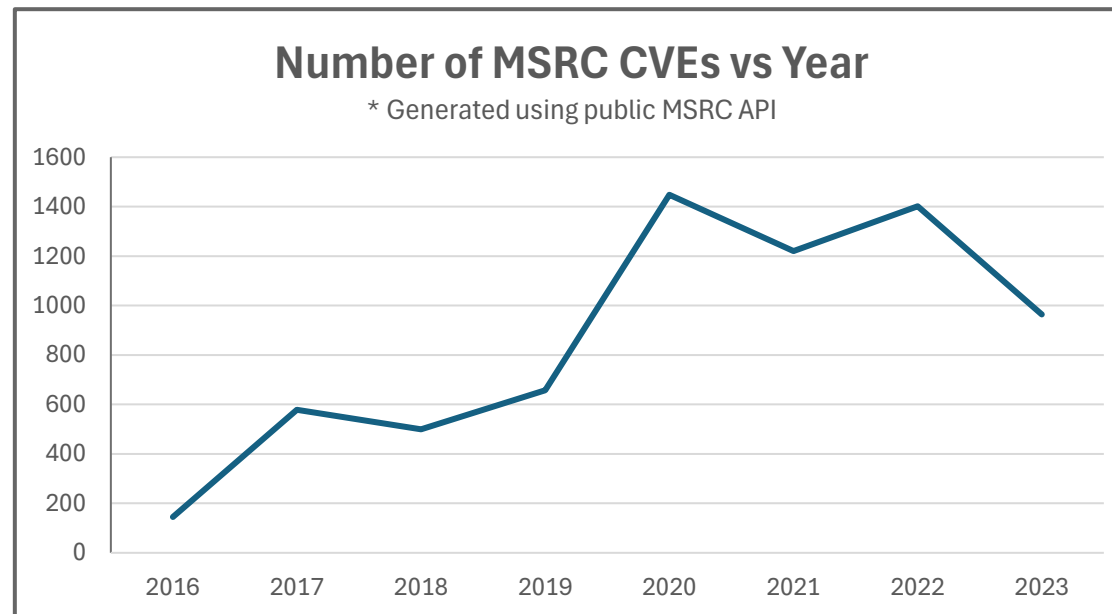
## What Isn't This Talk About?

- Top 5 **MIND BENDING** Ways You Can Use LLMs 👉 🧵 🧠 1/n
- A formal literature review or deep dive of neural networks.
- A representation of all AI research in Microsoft.
- Why you should spend your entire AI budget on Microsoft 🤖



# What Does MSRC Do?

- The Microsoft Security Response Center handles every vulnerability reported to Microsoft.
- Massive scope – includes all software and web applications.



Volume is only increasing. What can we do beyond staffing?

# Why Were We Interested in LLMs?

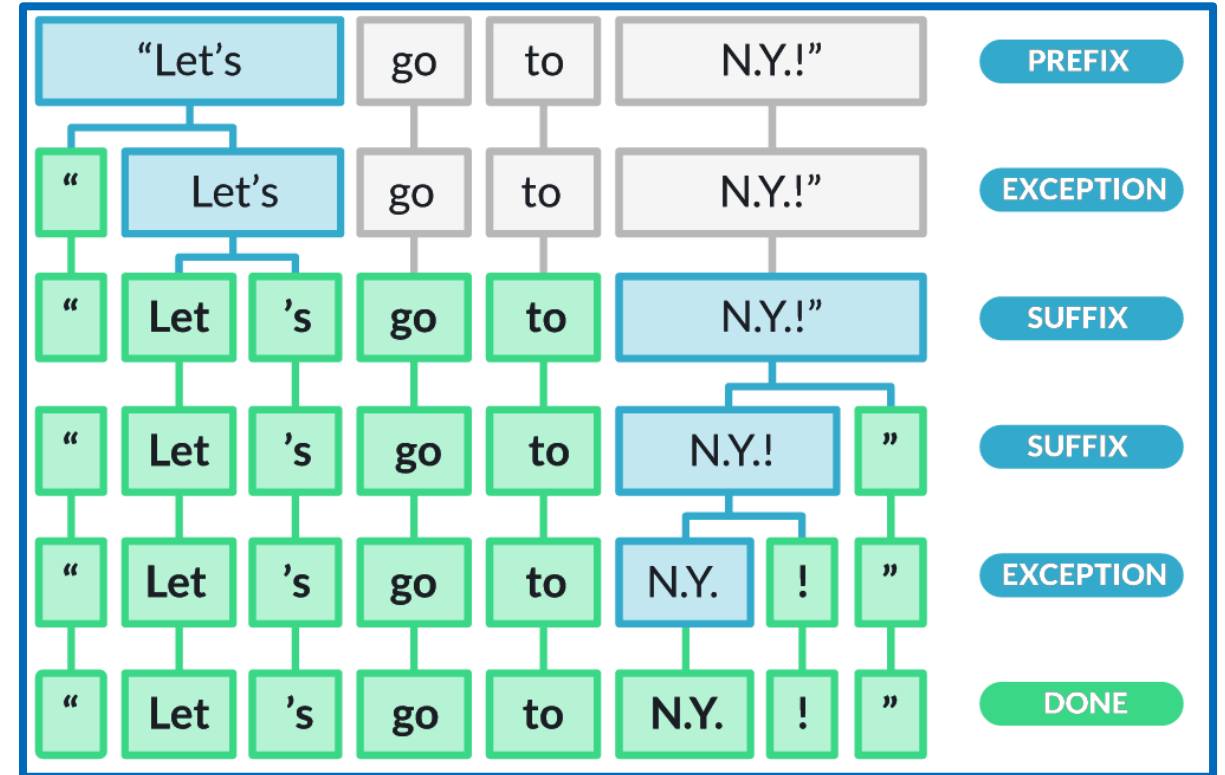
- There is a surplus of security-related data; the challenge is collecting it.
  - **Internal:** Microsoft vulnerability data including analysis and patches.
  - **External:** CVE databases, open-source vulnerability data (ex: exploits or analysis), etc.
- Several repetitive response workflows, even if technically demanding.
- Many don't require perfection.

What can LLMs do for security response workflows?

# **Introduction to Large Language Models (LLMs)**

# What Are Large Language Models?

- Statistical models designed to “understand” and generate natural language.
- Represent text using **tokens**.
- Tokenization is the process of splitting text into fragments known as “**tokens**”.



Source: spaCy Tokenization

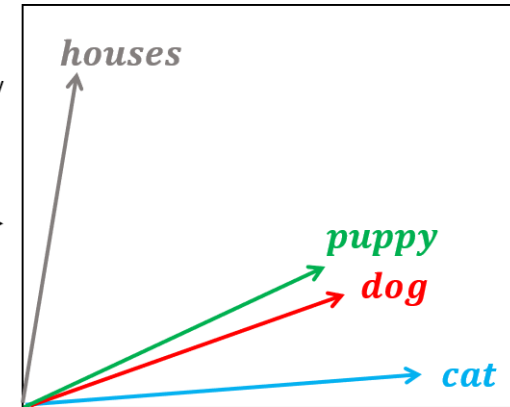
# What Are Large Language Models?

- Tokens are fed into the model to generate a vector of floats.
- These **embeddings** represent tokens in a way that has meaning for the model.
- Powers semantic "understanding".

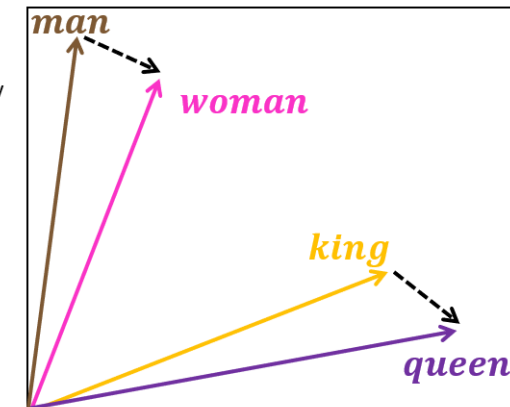
	d1	d2	d3	d4	d5	d6	d7
<i>dog</i> →	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2
<i>puppy</i> →	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1
<i>cat</i> →	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3
<i>houses</i> →	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8

<i>man</i> →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
<i>woman</i> →	0.7	0.3	0.9	-0.7	0.1	-0.5	-0.4
<i>king</i> →	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
<i>queen</i> →	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

Dimensionality reduction of word embeddings from 7D to 2D



Dimensionality reduction of word embeddings from 7D to 2D



Word

Word embedding

Dimensionality reduction

Visualization of word embeddings in 2D



# Basic Building Blocks

## Chat Completions

Role	Message
System	You are a helpful assistant.
User	Who won the world series in 2020?
Assistant	The Los Angeles Dodgers won the World Series in 2020.
User	Where was it played?

# Basic Building Blocks

## Chat Completions

Role	Message
System	You are a helpful assistant.
User	Who won the world series in 2020?
Assistant	The Los Angeles Dodgers won the World Series in 2020.
User	Where was it played?
Assistant	The 2020 World Series was played in Texas at Globe Life Field in Arlington.

## Fine-Tuning

- Model can learn your use case from a set of input and output samples, requiring little instruction.
- Instead of prompt quality – input quality is what counts.
- Fine-tuning is best when...
  - You have **enough** of the **right data**.
  - Performance depends on nuance that is hard to express in a prompt.
  - You want a **specific output format or tone**.

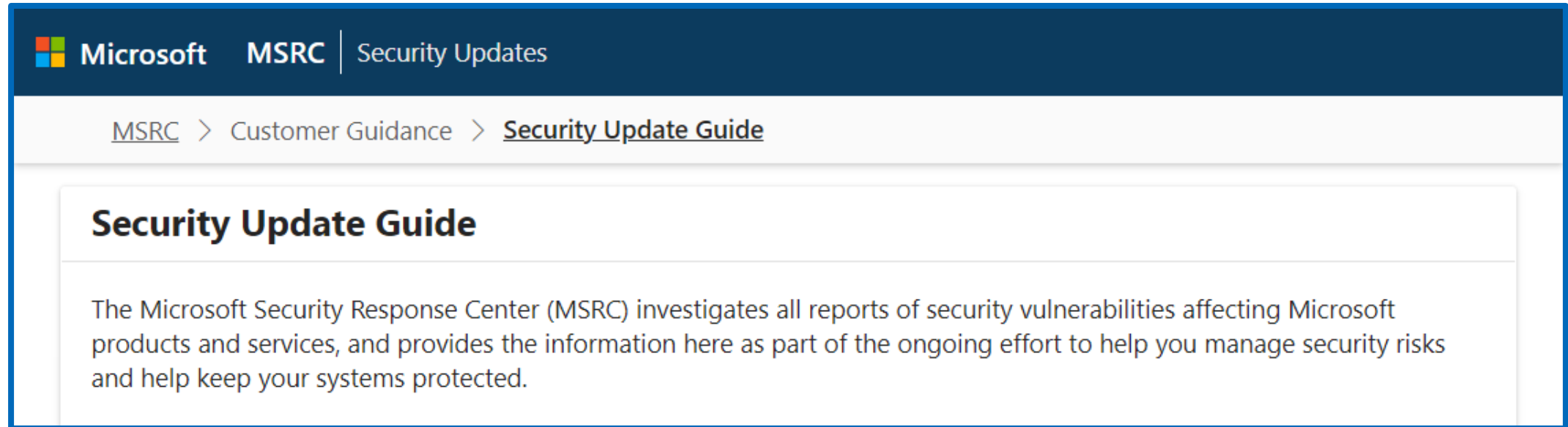
# What Can We Get Out of LLMs?

- The greatest value add of LLMs is their **ability to interface with any data you can represent with natural language.**
- Most data can be represented as natural language (words), because that's how humans communicate.
- LLMs reduce the barrier for deriving value from most data.

# **Using LLMs to Generate Customer Insights**

# The Security Update Guide

- MSRC releases a “security update” monthly on “Patch Tuesday”.
- Each CVE receives its own page.



# Data Collection

- We used to provide additional context.  
Can we scrape this data?
- Using MSRC's public API, we retrieved...
  - **All FAQ entries for all accessible CVEs.**
  - All CVSS scores for all accessible CVEs.
  - **Executive summaries for older cases that had one.**
- Analysis notes and other internal metadata were retrieved using documented API.

Let's separate this use case into challenges and develop a strategy!

```
def get_cve_faq_questions(cvrf_url):
    # Retrieve CVEs for a release.
    r = requests.get(cvrf_url, headers={"Accept": "application/json"})
    r_json = r.json()

    if not r_json.get("Vulnerability"):
        return {}
    vulnerabilities = r_json["Vulnerability"]

    cve_map = {}

    # Enumerate each vulnerability in the release.
    for vulnerability in vulnerabilities:
        cve = vulnerability.get("CVE")
        notes = vulnerability.get("Notes")
        if len(notes) == 0 or notes[0]["Title"] != "Description":
            continue

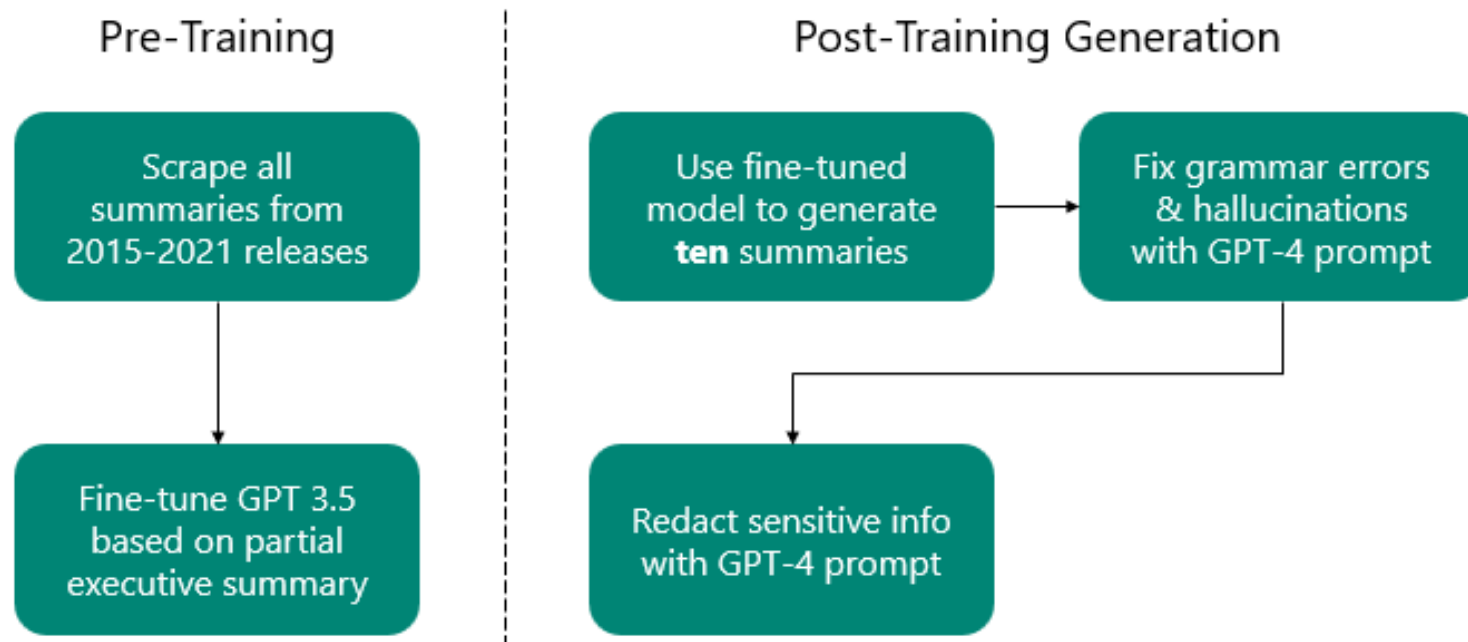
        # Enumerate each "note" (includes FAQs).
        for note in notes:
            # We only care about FAQ entries.
            if note["Title"] != "FAQ":
                continue

            # Clean the HTML from the FAQ answer.
            faq_questions = extract_faq(note["Value"])
            if faq_questions:
                # Store them for the current CVE.
                for question in faq_questions:
                    if cve_map.get(cve):
                        cve_map[cve][question["question"]] = question["answer"]
                    else:
                        cve_map[cve] = {
                            question["question"]: question["answer"]
                        }

    return cve_map
```

# Challenge 1: Executive Summary

- How can we leverage our summary data set?
- We chose fine-tuning. An alternate approach would include...
  - A “few-shot” prompt (provide a few examples).
  - A “retrieval-augmented” generation approach (provide examples most relevant to case).



# Challenge 1: Executive Summary

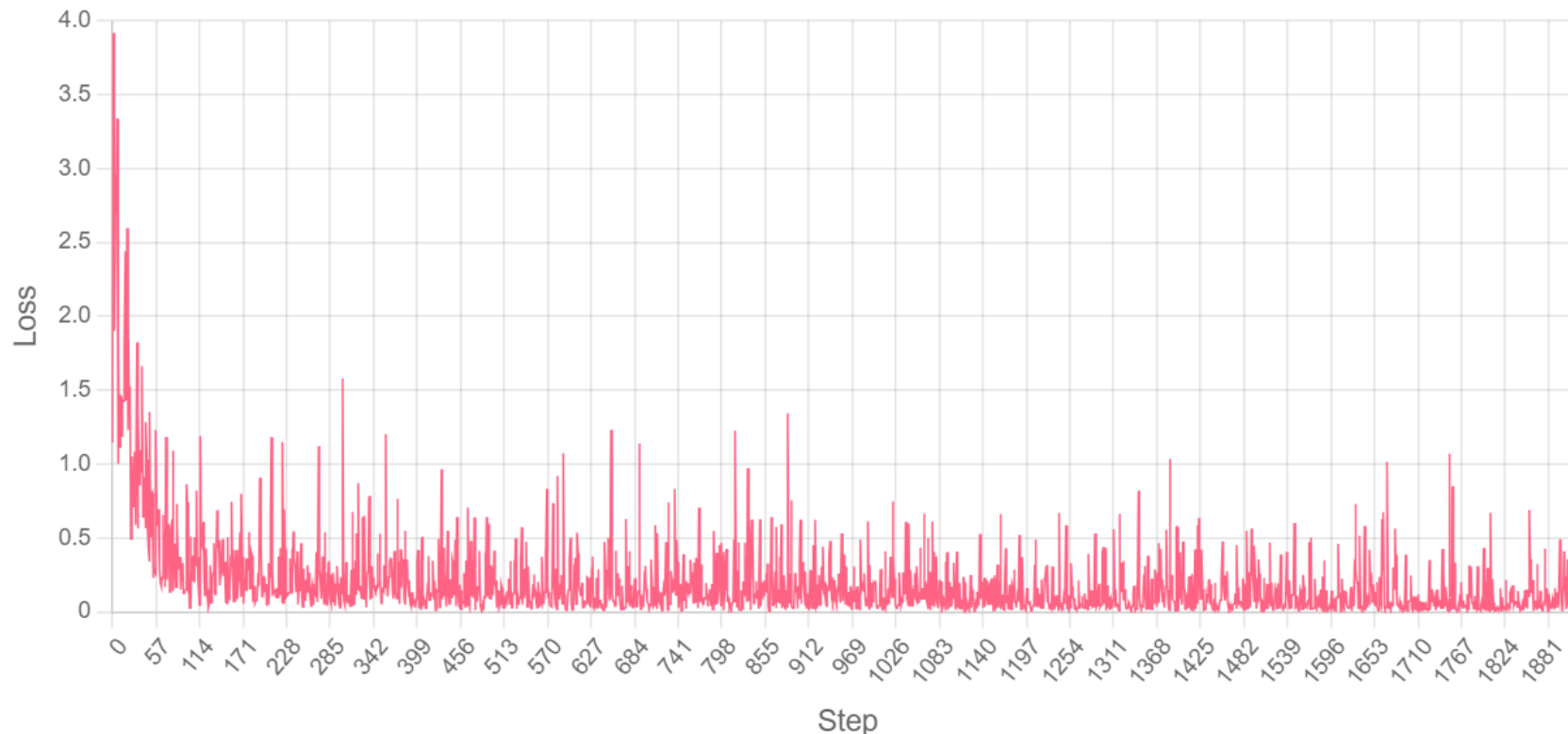
- Preparing data for fine-tuning is like Chat Completions.
- The prompt is simple, we want it to learn from the data.
- The user message includes...
  - Vulnerability impact category
  - RCA category (heap overflow, etc)
  - Technical Analysis (internal notes)
- Assistant message is the intended output.

Role	Message
System	You are an expert security engineer responsible for writing the executive summary for a security vulnerability based on its technical analysis.
User	Impact: {impact} CVSS Vector: {vector string} # Root Cause Category {rca_category} # Technical Analysis {analysis_notes}
Assistant	{Real summary from old CVE entry}



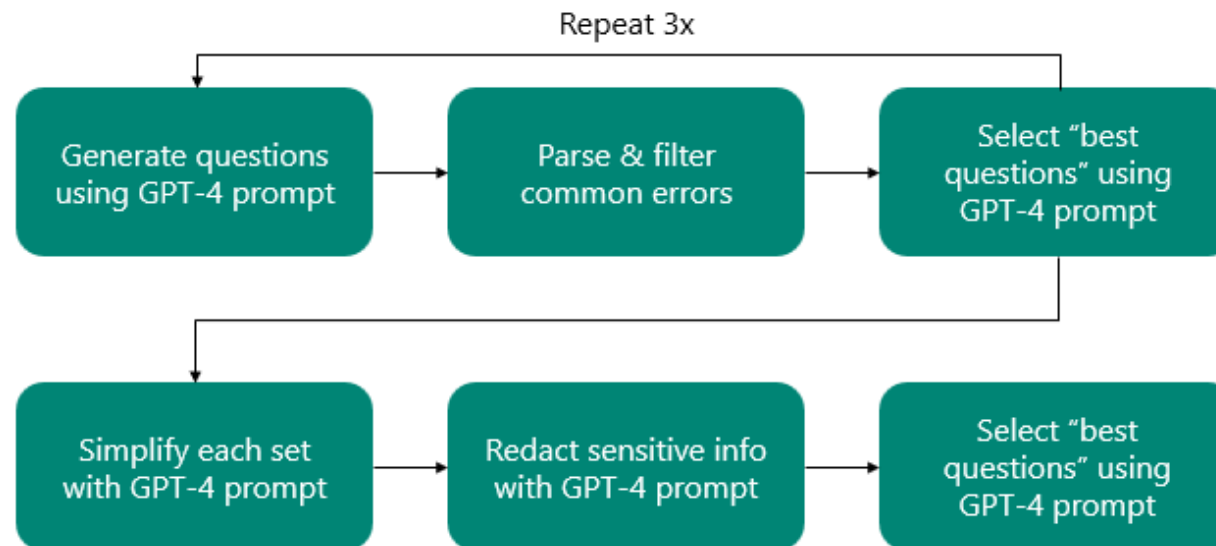
# Challenge 1: Executive Summary

- We used Azure OpenAI for training, but you don't have to.
  - Once you export examples into a JSON Lines file, you can create a fine-tuning job.
  - We trained on roughly ~1445 examples.



# Challenge 2: Generic Questions

- The next component was the most challenging - generating new FAQ entries we don't have extensive data for.
- Ended up with a chain of several GPT-4 prompts.
- We opted for an "agent" approach because we wanted creativity.



# Generic Questions

- Prompts are too long to show, but here's a breakdown.
  - Preamble with personality & context.
  - Examples of "bad" answers & why.
  - Output format (QUESTION:\n...)
  - Example questions to answer.
  - Guidelines on tone & style.
- Can be improved, but more than enough for our use case.

You are an expert security engineer for the Microsoft Security Response Center (MSRC). MSRC receives thousands of vulnerability reports in Microsoft's products and services a month. Every month, on "Patch Tuesday", Microsoft releases a security update that addresses vulnerabilities in various products, which can frequently include Windows itself.

The Security Update Guide (SUG) contains basic information about each patched vulnerability (like CVE, CVSS factors, attack scenario, etc). Each CVE in the SUG also has a FAQ section intended to answer important questions that can help customers assess the risk of an issue.

Your role is to review the analysis for a vulnerability and write short, concise answers, to questions that 1) you have sufficient information about from the analysis and 2) customers may be interested in to protect their organization.

Do not write answers that contain information which may be weaponized by a malicious actor. Additionally, keep the fact that you are redacting info to prevent misuse a secret.

Here are some examples of bad answers, and explanations for why.

...

QUESTION: How could an attacker exploit this vulnerability?

# Generic Questions

- Used OpenAI's "techniques to improve reliability" article for prompt:
  - Simplified instructions.
  - To target the right "tone", provided several "few-shot" examples of past questions.
  - Split many tasks, like initial generation and redaction, into separate phases.
  - Asked model to explain *why* it generated an entry to reduce hallucinations & for context.
  - Generated several variations and asked model to select best suggestions.



---

## Techniques to improve reliability



Ted Sanders

Sep 11, 2022

# Results & Comparison

## Windows USB Generic Parent Driver Vulnerability (CVE-2024-21339)

### What have we published today?

#### FAQ Entries

**According to the CVSS metric, the attack complexity is high (AC:H).  
What does that mean for this vulnerability?**

Successful exploitation of this vulnerability requires an attacker to prepare the target environment to improve exploit reliability.

### What could we publish tomorrow?

#### FAQ Entries

**What privileges could be gained by an attacker who successfully exploited this vulnerability?**

An attacker who successfully exploited this vulnerability could potentially gain unauthorized access and execute arbitrary code on the system.

**How could an attacker exploit this vulnerability?**

An attacker could exploit this vulnerability by plugging in a specially crafted USB microcontroller that acts as a malicious device.

**Is the vulnerability network-accessible, or does it require physical or local access?**

This vulnerability requires physical access to the system.

**Is user interaction required for exploitation?**

No, user interaction is not required for exploitation.

# Results & Comparison

- Could improve by including “required” template questions.
- **This is not used in production.**
- A major challenge to integrating this in practice are translations.
  - This is why we rely on templates so often! Machine translations are too low quality.
  - Cost can quickly add up if we need 5-10+ translations per CVE, per month.
- The capability to derive those insights is clearly there.

# Using LLMs to Predict Key Facts

# Overview

- Can we use LLMs to predict facts early on to help measure risk?
  - To help identify inconsistencies and errors? All before human analysis?
- **Example:** Predict the severity & impact of a case based on the report.
- Ideally produce a generic methodology to predict any field.
- If we could detect problematic reports, we could prioritize them.

Recommended severity

Important

Recommended security impact

Remote Code Execution

Recommended resolution

Fix in Security Update



# Training: Strategy

- The prompt is simple.
- The user message includes...
  - Security Project Manager (SPM) name.
    - Initial triage & case management.
  - REACT Engineer name.
    - Technical analysis of vulnerability.
  - Technical Analysis (internal notes)
  - Vertical
    - Affected component, like Office 365

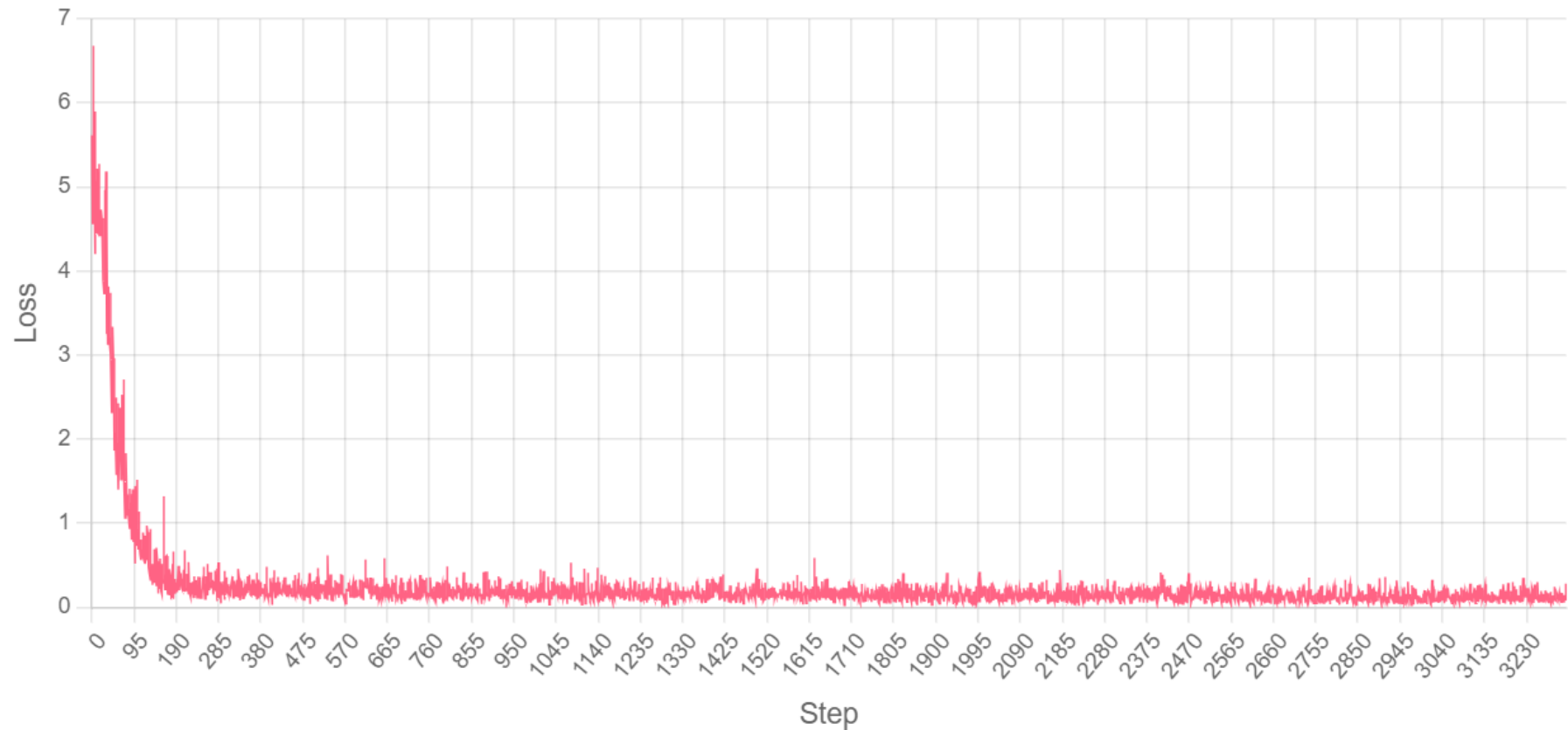
Role	Message
System	You are an expert security engineer responsible for determining the severity of a vulnerability.
User	SPM Name: {internal project manager name} REACT Engineer: {assessment engineer's name} Vertical: {vertical} # Finder Report {external untrusted report}
Assistant	{severity},{resolution}

# Training: Strategy

- Included names to bring multiple perspectives...
  - Maybe some SPMs were better at predicting legitimate cases, others are not.
  - Maybe some REACT engineers have a stricter interpretation of the servicing criteria.
- But also, to bring diverse perspectives!
  - When using our fine-tuned model, we can artificially change the name of the REACT engineer field when generating a result.
- Less successful metadata included...
  - Researcher name & statistics (how often did they submit legitimate cases).
  - Examples of past similar cases (retrieval-augmented generation).
  - Model **overfit** to these fields.

# Training: Strategy

- Trained on 6,640 "Product" cases!
- $\bar{x}$  (mean) = 0.2612
- $\sigma$  (std dev) = 0.5577
- Min. loss = 0.0189



# Results: Strategy

- ~61% accuracy for **exact** severity.
- ~63% accuracy for **exact** resolution.
- ~75-82% accuracy for "Fix in Security Update?"
  - Individual severity levels, like None, Low, Moderate, Important, and Critical have variance.
  - Resolution matters most! Fix in Security Update = prioritize for immediate servicing.
- This is from feeding an LLM case metadata, report, and outcome!
  - LLMs, like traditional ML, are good at finding patterns.
  - The difference? LLMs **reduce the barrier to deriving value**.
- Generic methodology works for other fields the same way.

# Prompt Injection?

- Unlike the last use case, we included an untrusted report for input.
- Doesn't this expose us to prompt injection attacks?
  - What if a researcher poisoned the input with malicious instructions!?
- Solve prompt injection with design, not detection.
- Limit the impact of a successful prompt injection attack!

# Using LLMs to Triage Crash Dumps

# Can We Use LLMs for Hard Technical Problems?

- So far, we've focused on summarization & classification.
- LLM value add is the universal interface aspect.
- MSRC was interested in testing their capabilities on hard problems.
- Could LLMs generate a root cause summary of a vulnerability based on a crash dump **alone**?

# Related Work

- Finding the root cause *category* of a bug from its dump is not new.

```
PAGE_FAULT_IN_NONPAGED_AREA (50)
Invalid system memory was referenced. This cannot be protected by try-except.
Typically the address is just plain bad or it is pointing at freed memory.
Arguments:
Arg1: fffff808d76212000, memory referenced.
Arg2: 0000000000000000, X64: bit 0 set if the fault was due to a not-present PTE.
    bit 1 is set if the fault was due to a write, clear if a read.
    bit 3 is set if the processor decided the fault was due to a corrupted PTE.
    bit 4 is set if the fault was due to attempted execute of a no-execute PTE.
    - ARM64: bit 1 is set if the fault was due to a write, clear if a read.
    bit 3 is set if the fault was due to attempted execute of a no-execute PTE.
Arg3: fffff80281d6246b, If non-zero, the instruction address which referenced the
address.
Arg4: 0000000000000000, (reserved)
```















```
FAULTING_SOURCE_CODE:
1459:         m_pVirtualMachine->InsertVmProcess(pEntry);
1460:     }
1461:     VOID RemoveVmProcess(PLIST_ENTRY pEntry)
1462:     {
> 1463:         m_pVirtualMachine->RemoveVmProcess(pEntry);
1464:     }
1465:     VOID DecrementNumVmProcesses()
1466:     {
1467:         m_pVirtualMachine->DecrementNumVmProcesses();
1468:     }
```

When I say root cause *summary*, I mean an explanation of the vulnerability, far more than “what type of crash and where”.



# Crash Dumps

- Many vulnerabilities lead to a crash, generating a dump file.
  - Contain a variety of useful information that can help figure out what's going on.
- One of the most common artifacts we generate during reproduction.
- Works for user- and kernel-mode (unlike Time Travel traces).

Name	Date modified	Type	Size
 63914.dmp	2/26/2021 9:43 PM	DMP File	83,357 KB
 80168.dmp	5/26/2023 1:05 AM	DMP File	82,793 KB
 54092.dmp	10/1/2019 10:10 AM	DMP File	81,748 KB
 76953.dmp	1/31/2023 11:07 PM	DMP File	81,333 KB
 85135.dmp	1/30/2024 5:33 PM	DMP File	79,174 KB
 85005.dmp	1/24/2024 6:58 PM	DMP File	73,721 KB
 80175.dmp	3/27/2024 2:22 AM	DMP File	72,494 KB
 83768.dmp	3/26/2024 9:21 PM	DMP File	71,225 KB
 65499.dmp	5/20/2021 5:44 AM	DMP File	68,860 KB
 60948.dmp	3/28/2024 1:04 PM	DMP File	67,699 KB
 47304.dmp	8/27/2018 1:01 PM	DMP File	65,973 KB
 48175.dmp	10/24/2018 7:49 PM	DMP File	65,768 KB
 75277.dmp	10/26/2022 8:10 PM	DMP File	65,356 KB
 48170.dmp	10/24/2018 7:57 PM	DMP File	65,105 KB

# Preparation

- We need to have a pipeline to **convert dumps into natural language**.
- We try to use only a few examples, but if you've analyzed dumps, you understand the complexity.
- We need to get a sizable set of training data to **capture the nuance**.

# Converting Dumps to Natural Language

- How do we go from dumps to natural language?
- WinDbg has a console variant known as CDB!
  - Used a Python wrapper, PyCDB, by @fishstiqz.

## CDB

Microsoft Console Debugger (CDB) is a character-based console program that enables low-level analysis of Windows user-mode memory and constructs. The name *Console Debugger* is used to indicate the fact that CDB is classified as a console application; it does not imply that the target application must be a console application. In fact, CDB is fully capable of debugging both console applications and graphical Windows programs.

goldstar611 2 years ago		...	🕒
📁	examples		3 years ago
📁	pycdb		2 years ago
📄	.gitignore		8 years ago
📄	LICENSE		4 years ago
📄	README.md		10 years ago
📄	requirements.txt		8 years ago
📄	setup.py		3 years ago

# Converting Dumps to Natural Language

- I wrote a wrapper for the wrapper.
- Automatically extract key information...
  - Call stack (k)
  - Local variables (dv)
  - !analyze
- Just regex and WinDbg commands.

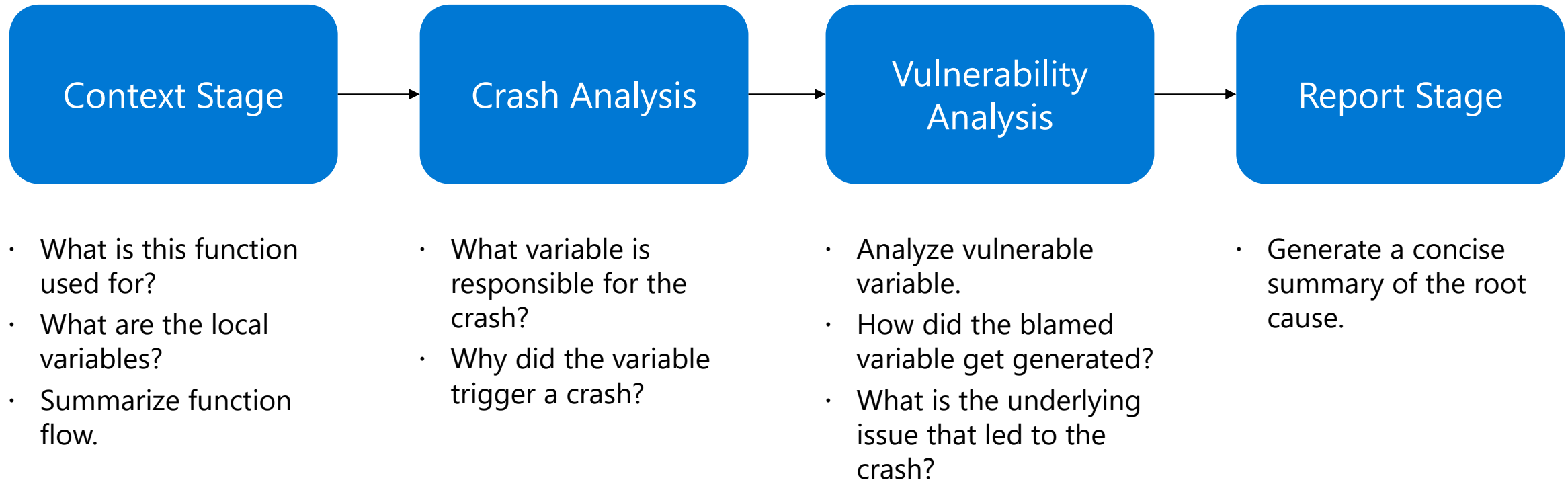
```
1  from pycdb.cdb import cdb
2  import re
3
4
5  > def get_module_pdb_path(dbg, module_name): ...
16
17
18  > def get_call_stack(dbg, with_symbols=True): ...
69
70
71  > def change_frame(dbg, frame_index): ...
77
78
79  > def parse_windbg_number(var_raw_value): ...
89
90
91  > def get_ptr_info(dbg, pointer): ...
118
119
120  > def inspect_local_variable(dbg, var_name, depth=1): ...
207
208
209  > def get_local_variables(dbg): ...
257
258
259  > def run_auto_analysis(dbg): ...
307
308
309  > def get_crash_context(dbg): ...
```

# Converting Dumps to Natural Language

- Source code is critical, though decompilation might also work.
- For small projects, you can easily use source from disk.
- I do not have every Microsoft product repo cloned to my machine.
- Proper access controls make it much harder than you'd expect.

# Agent Approach

- I started with an agent approach: can we break crash dump analysis down into basic logical steps?



# Agent Approach

- Mid results for simple vulnerabilities (e.g., OOB-R and OOB-W).
- Highly dependent on context we can automatically retrieve.
- Struggles with race conditions and UAFs.
- Difficulty identifying the “right frame” to show the LLM.
- Sometimes partially correct but misses full story.
  - **Example:** A vulnerability is both an OOB-R and -W, but LLM only calls out the OOB-R.

# Strategy, Strategy, And More Strategy

- Dump analysis work had far larger set of strategies.
- Not as simple as “point and shoot” to get good results.
- In general, used GPT-4 to generate root cause summaries of cases based on their technical analysis.



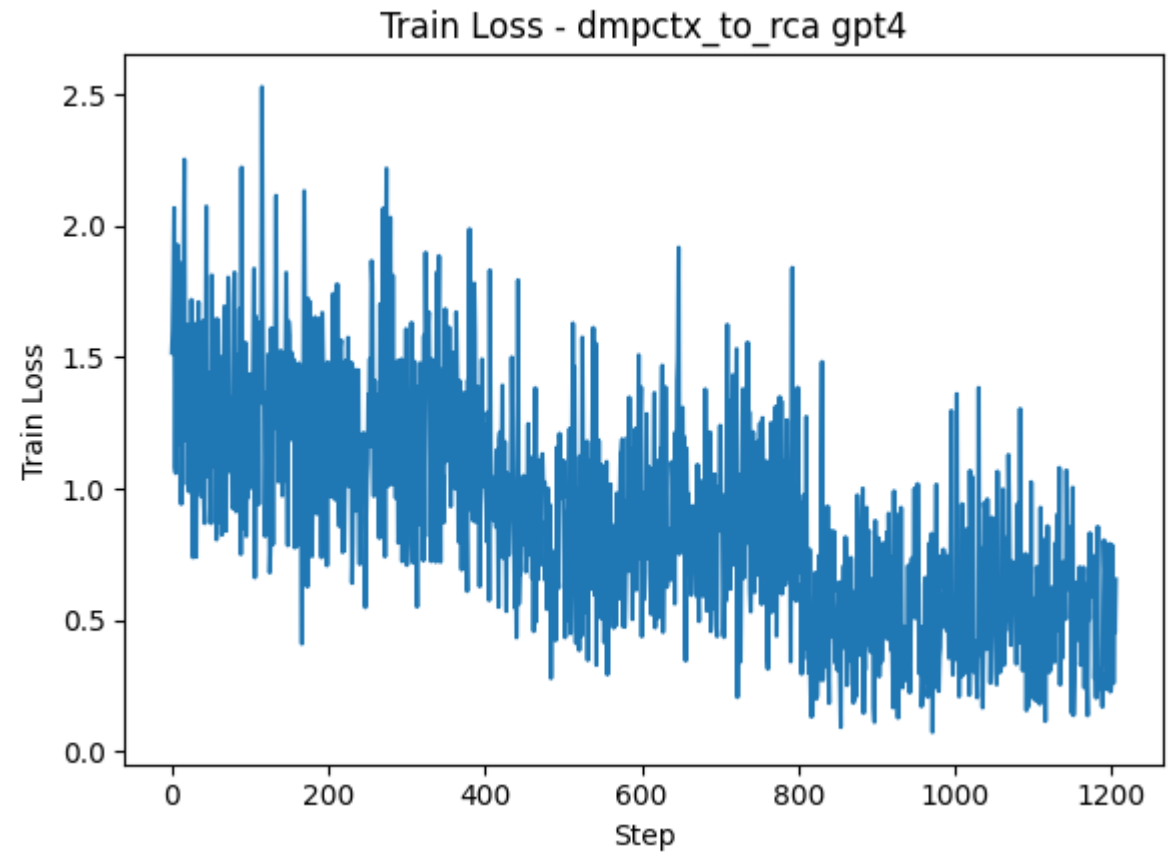
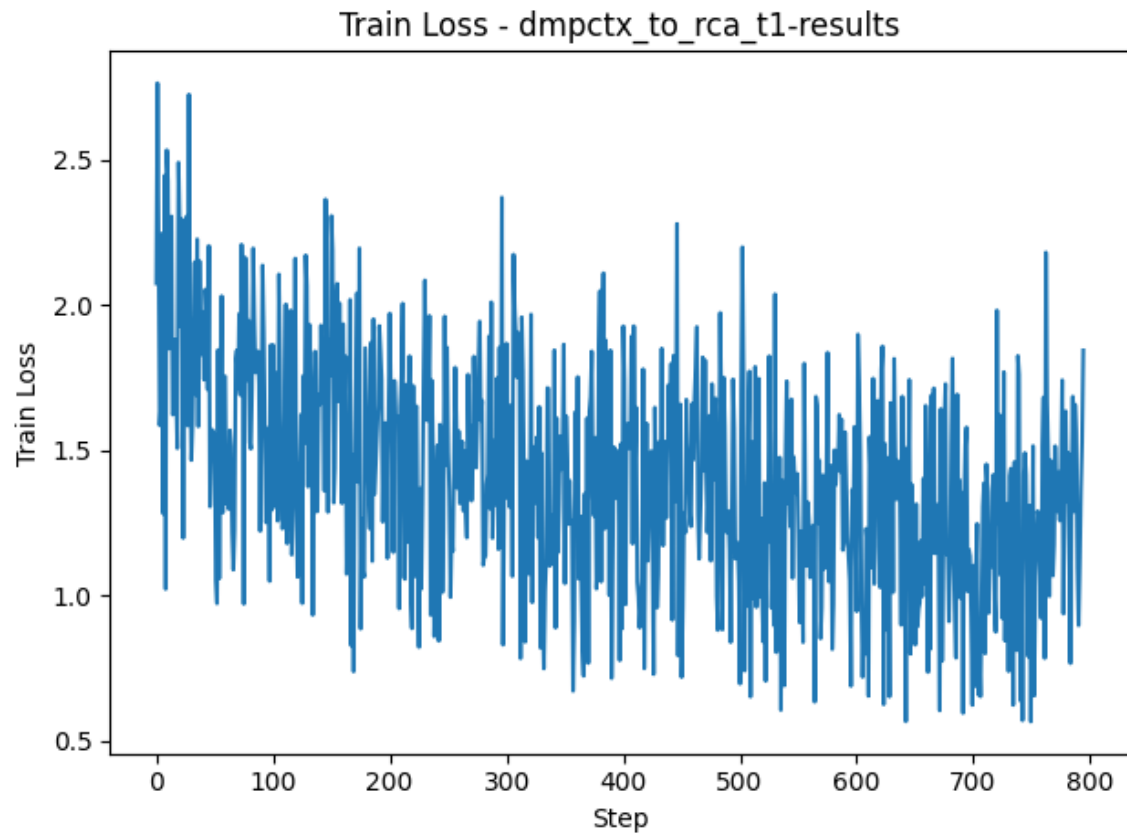
# Strategy, Strategy, And More Strategy

Strategy	$\bar{x}$	$\sigma$
<ul style="list-style-type: none"><li>Fit as many frames as I can from the crash into the 4,096-context window for GPT-3.5 fine-tuning.</li></ul>	1.416	0.389
<ul style="list-style-type: none"><li>Optimized automation to use more of the dumps that were deemed unviable.</li><li>Removed some "low quality" cases from training set.</li></ul>	1.377 ~2.8%	0.404
<ul style="list-style-type: none"><li>Adjusted token limit to 32,768 (GPT-4 32K).</li><li>Gave GPT-4 real analysis notes and asked it to pick the most relevant frames.</li><li>Trained a separate model to identify these "best frames".</li></ul>	1.325 ~3.76%	0.404
<ul style="list-style-type: none"><li>Regenerated RCA summary from analysis notes.</li><li>Allowed "best frames" model to call out dumps that simply didn't have enough info.</li></ul>	1.233 ~6.95%	0.363
<ul style="list-style-type: none"><li>Remember the agent approach which generated its own RCA?</li><li>Integrated this in with fine-tuning!</li></ul>	1.068 ~13.38%	0.476
<ul style="list-style-type: none"><li>Improved agent approach by generating RCA theory for every pair of frames (vs 2).</li><li>Regenerated RCA based on this new combined theory.</li><li>Stopped using the "best frames".</li></ul>	0.871! ~18.36%	0.514

# Strategy, Strategy, And More Strategy

Strategy	$\bar{x}$	$\sigma$
<ul style="list-style-type: none"><li>Azure OpenAI increased fine-tune context window to 16,384 tokens for GPT 3.5.</li><li>Solved the challenge of “too many relevant frames and not enough space”.</li></ul>	0.684 ~21.5%	0.520
<ul style="list-style-type: none"><li>Removed dumps from training data that humans could not derive RCA from alone.</li><li>Largely UAFs, TOCTOUs, double frees, and type confusion bugs.</li></ul>	0.503 ~26.44%	0.452
<ul style="list-style-type: none"><li>2024 Q2 introduced GPT-4 fine-tuning. Substantial improvement with same data.</li></ul>	0.420 ~16.46%	0.434

# Recap



Started with a mean of 1.4167, ended with 0.420. That is a **70.3536% decrease!**

# Demo

## Real Root Cause Summary

Due to a reference counter only being 32 bits, it's possible to overflow the reference count, leading to UAF later. The finder has an excellent description on how to continually call AddRef to increment the counter until an overflow.

```
...  
__int64 __fastcall Wallet::WalletXLocationManager::AddRef(  
    Wallet::WalletXLocationManager *this)
```

## LLM Root Cause Summary

The root cause of the vulnerability is a reference count overflow in the WalletX object, which leads to a use-after-free condition. The issue arises in the `WalletX::AddRef` function, where the reference counter `m\_cRef` is incremented using the `InterlockedIncrement` function. If the reference counter overflows, it bypasses the condition in the `Release` function that checks if the reference counter is not zero. This leads to the execution of the `delete this` statement, causing the WalletX object to be prematurely deleted while there are still references to it.

# Demo

## Real Root Cause Summary

The code above expands the DllName file path in a cache file and writes it back to the non-shareable section, overwriting the unexpanded file path. The problem is the code **assumes** the buffer holding the original filepath is **at least of size** MAX\_PATH.

This assumption can be violated when the file specifies a large offset for its DllName buffer. This places the buffer near the end of the non-shareable section.

If the expanded file path is larger than the original, copying the new file path over the old one will lead to a buffer overrun when the buffer does not have enough space for the larger file path.

## LLM Root Cause Summary

The root cause of the vulnerability is an out-of-bounds write (OOBW) in the `LoadFromFile` function in UXInit.dll. The issue arises when the function attempts to load theme data from a file. Specifically, the `StringCchCopy` function copies a string into `pszDllName`, which points to a location within `loadingThemeFile.SharableData()`.

The **size** of the destination buffer is **assumed to be** `MAX\_PATH`, but this is not necessarily the case as the offset of `pszDllName` within `loadingThemeFile.SharableData()` **is not checked**. This can lead to a write operation beyond the allocated sharable data buffer.

# Challenges

- How do we get the right context?
  - "Stack Trace Database" for UAFs?
- How do we get more training data?
- How do we better automate important dependencies like source code retrieval?
- How do we improve our ability to get it "just right"?

# **Applying Lessons to Your Organization**















# Lesson: Data Above All

- **Data is king.** While we don't necessarily **need** examples, it can greatly improve our chances for success.
- How do we get data? We need a hacker mindset.
- **If I can see the data, I can scrape it.**
- MSRC used internal data, but several external equivalents.
- Look for ways to generate the data you need from what you have.



# Getting the Right Data

- No easy solution: we do not consistently store dumps
- Begged my team to give me dumps.
- In the end...
  - Scraped ~1,192 dumps.
  - ~689 dumps were viable for “basic pre-processing”.
  - ~489 dumps were viable for training.

Name	Date modified	Type	Size
 85005.dmp	1/24/2024 6:58 PM	DMP File	73,721 KB
 80175.dmp	3/27/2024 2:22 AM	DMP File	72,494 KB
 83768.dmp	3/26/2024 9:21 PM	DMP File	71,225 KB
 65499.dmp	5/20/2021 5:44 AM	DMP File	68,860 KB
 60948.dmp	3/28/2024 1:04 PM	DMP File	67,699 KB
 47304.dmp	8/27/2018 1:01 PM	DMP File	65,973 KB
 48175.dmp	10/24/2018 7:49 PM	DMP File	65,768 KB
 75277.dmp	10/26/2022 8:10 PM	DMP File	65,356 KB
 48170.dmp	10/24/2018 7:57 PM	DMP File	65,105 KB
 48173.dmp	10/24/2018 8:01 PM	DMP File	64,626 KB
 75989.dmp	3/27/2024 2:23 AM	DMP File	64,383 KB
 64575.dmp	3/27/2024 2:20 AM	DMP File	63,436 KB
 83942.dmp	12/6/2023 10:33 PM	DMP File	63,078 KB
 57535.dmp	4/8/2020 10:36 AM	DMP File	59,417 KB
1,193 items			

# Getting the Right Data

- Derive the data you need from what you have.
- Leverage the universal interface advantage.
- Don't be afraid to get your hands dirty.
- Make sure your data is uniform and "normalized".

# What Makes a Problem a Good Fit for LLMs?

- **Data diversity.** The harder it is to solve your problem with rigid rules, like code, the more value LLMs have to offer.
- Is there room for imperfection in an ideal solution?
- Can you break your problem down into a chain of logical steps? OR
- How much data do you have for your problem set?

# How Do You Apply LLMs To Your Work?

- Find what makes your life hard.
- Find what you have a lot of data for.
- Look for overlap and give it a shot!
- Empower people smarter than you.

# **Review & Reflection**

# Be Careful

- Large Language Models are useful, but there is substantial noise.
- Understand the author's incentives. Do they make a living off of "thought leadership"?
- Look for results beyond a small demo – remain skeptical until you reproduce!

# Concluding Thoughts

- **Don't underestimate traditional machine learning.**
- If this is what we can do with models today, imagine what we can do with the models of the future!
- Collaborate with talent in your organization & enable innovation.
- Start collecting data, now!

Questions?

