A TENABLE RESEARCH REPORT

HOW GENERATIVE AI IS CHANGING SECURITY RESEARCH

FOREWORD

By Ray Carney, Director, Security Response and Zero-Day Research, Tenable

If they can get you asking the wrong questions, they don't have to worry about the answers. – Thomas Pynchon, "Gravity's Rainbow"

This quote from Thomas Pynchon can probably be boiled down to two words – cognitive bias. That's just a fancy way of saying mental model, or mindset.

The question at hand is this: What is a generative pre-trained transformer (GPT) or generative artificial intelligence (AI)? Let's start with a baseline. Generally speaking, GPT is a form of generative AI. More specifically, it is an AI algorithm that generates outputs based upon the data it has been trained on. It's a type of machine learning based upon a large language model (LLM) and intended to emulate a human response. It does this by leveraging an extremely large corpus of data to train the model to predict the next word. In simplest terms, a well-trained GPT model would be expected to generate an output indiscernible from a human response based upon a given prompt or question.

As it turns out, this is actually pretty hard, even for the easier, more mundane prompts. But as you move right on the continuum from mundane to specialized, the difficulty increases drastically. It's so difficult, in fact, that OpenAI included the following statement in the announcement of its GPT-4 release regarding limitations:

"Despite its capabilities, GPT-4 has similar limitations as earlier GPT models. Most importantly, it still is not fully reliable (it 'hallucinates' facts and makes reasoning errors). Great care should be taken when using language model outputs, particularly in high-stakes contexts, with the exact protocol (such as human review, grounding with additional context, or avoiding high-stakes uses altogether) matching the needs of a specific use-case."[1]

What OpenAI is saying is that the model, much like a human, has cognitive biases. These biases are a result of several factors, including:

- the model's "experiences";
- the way the model was trained;
- the incomplete and imperfect information in the corpus of data used to train the model; and
- the cognitive biases of the developers of the model itself.

All of the above become intrinsic elements of the AI's mental model and constitute the model's cognitive biases.

[1] https://openai.com/research/gpt-4

On top of all that, we also have to consider the cognitive biases of the person or program that is querying the model. This is the really important part because once you input enough information specific to a given domain of knowledge into a model like GPT-4, asking the right questions becomes the most critical factor in the probability of producing a correct answer. And, as we have already established, "If they can get you asking the wrong questions, they don't have to worry about the answers."

Ultimately, the role of a security analyst is to provide timely and accurate data to a decision maker. In pursuit of this goal, the analyst must process and interpret collections of incomplete and ambiguous data in order to produce sound, well-founded analytical judgments. Over the course of many years, and many failures, the analytical community has developed a set of tools commonly referred to as 'structured analytic techniques' that help to mitigate and minimize the risk of being wrong, and avoid ill-informed decisions.

The warnings posed by OpenAI in its announcement of GPT-4 make a strong argument for the application of these techniques. In fact, it is only through the application of these types of techniques that we will ultimately produce a well-refined dataset to train future models in the cybersecurity domain. These types of techniques will also help researchers to ensure that they are tuning their prompts for those models — that they're asking the right questions. In the meantime, security researchers can continue to investigate how we leverage generative AI capabilities for more mundane tasks in order to free up time for researchers and analysts to invest their time on the more difficult questions that require their subject matter expertise to tease out critical context.

The following paper delves into some early examples of how Tenable Research is experimenting with the use of LLMs to reduce complexity and achieve efficiency in the following areas in order to accelerate research capabilities:

- Reverse engineering
- Debugging code
- Improving web app security
- Increasing visibility into cloud-based tools

While this is early days for this type of experimentation, these tools have been made available to the wider community via a GitHub repository for further iteration and adoption. Tenable will continue to test such technologies to help organizations look for new ways to reduce their cyber risk.

INTRODUCTION

The cybersecurity experts on the Tenable Research team continuously monitor the threat landscape and examine the shifts in technologies and techniques employed by attackers. The work involves analyzing enormous amounts of data in order to provide contextually relevant guidance to the organizations using our products.

In recent months, the security community — as well as government agencies worldwide — have raised concerns about the use of AI technologies like OpenAI's ChatGPT. While the organizations developing these tools have demonstrated impressive advancements in generative AI, many in the security community recommend proceeding with caution.

These tools, built on LLMs, are evolving at breakneck speed. With their ability to produce human-like responses, LLMs provide ample opportunities for unique use cases. Popular applications — such as chatbots which can reduce workloads and lead to faster response times for customers — are having a major impact in the customer service industry. However, these advanced models also face scrutiny by the security community in anticipation of use by attackers.

At Tenable Research, we take seriously the concerns of the broader security community — and we also see enormous potential for the use of LLMs to aid in both offensive and defensive cybersecurity. Our researchers have embraced these technologies, examining how LLMs can be used in both offensive and defensive capabilities, and we're pleased to share the results of our work in this report.

HERE BE MONSTERS?

At the present time, there are more questions than answers about the possible dangers generative AI poses in cybersecurity. Attackers are opportunistic and, in many cases, financially motivated, seeking the path of least resistance to breach organizations by any means. While AI could be used to streamline malware development and craft phishing messages, we have yet to see evidence of a threat actor effectively using these tools in a way that raises the bar for attackers. Yet, many researchers have raised concerns that LLMs could be abused as writing assistants to forge phishing emails and other props for social engineering attacks and scams. It's possible that these tools will provide attackers with more convincing phishing emails by training models to use specific language which can be taken from a targeted organization's website, press releases or social media accounts. While advanced LLMs are showing promising results, they are still in the very early stages of development and are prone to mistakes that in many instances may be easily identified.

LLMs and the applications that use them may also become tempting targets for attackers seeking to influence their behavior through prompt injections or to extract sensitive data embedded in the models and their prompts. For these and other reasons, we're seeing calls to slow the development of AI systems. There's tremendous interest in making sure AI technology is responsibly developed, with organizations like the National Institute of Standards and Technology (NIST) launching the Trustworthy & Responsible Artificial Intelligence Resource Center (AIRC).

While there's certainly a dark side to these emerging technologies, Tenable Research also sees opportunities to use AI for the greater good. For example, the art of bug hunting requires extensive security and coding skills and it can take years for an individual to develop the necessary expertise to find zero-day vulnerabilities. As researchers we turn our mindsets to using and developing tools that can reduce manual labor. With these generative models, we have a unique opportunity to change the trajectory of security research.

HOW THE TENABLE RESEARCH TEAM IS USING LLM-POWERED TOOLS

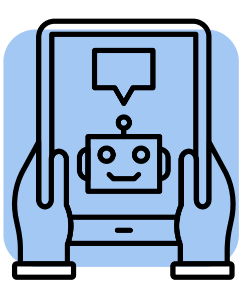
As a research group, our team needs to think like attackers and adopt the hacker mindset. With that in mind, we set our sights on the possible uses of AI in security research. We found such tools offered novel ways to perform security research in the domains of reverse engineering and vulnerability analysis. While these tools are far from replacing security engineers, they can act as a force multiplier and reduce some labor-intensive and complex work when used by experienced researchers. While we've only just scratched the surface of how AI can play a role in security research, Tenable has already used LLMs to build new tools that are speeding our processes and helping us identify vulnerabilities faster and more efficiently. Below, we highlight four ways our team is benefiting from generative AI. We're paying it forward by making these tools available to the research community.

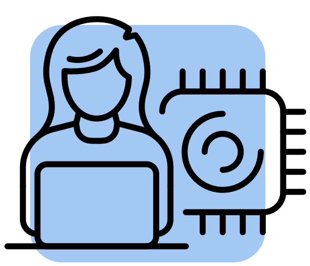
G-3PO: A Translation Script For Ghidra

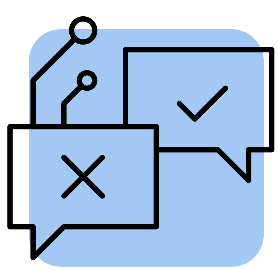
Reverse engineering requires extensive knowledge and practice to master. But what if we could lower the barrier to entry or reduce some of the complexities involved as well as aid a seasoned researcher using the power of AI?

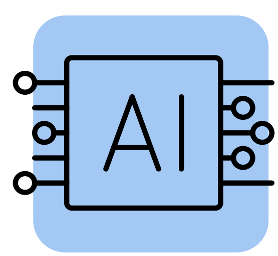
As researchers, the right tools and techniques make all the difference and one of the favorite reverse engineering tools amongst the community is Ghidra, an extensible software reverse engineering framework developed for internal use by the NSA. It was declassified and released to the general public in 2019, quickly becoming a favorite tool in the security community. It handily automates several reverse engineering tasks, including disassembling a binary into its assembly language listing, reconstructing its control flow graph and decompiling that assembly listing into something resembling source code in the C programming language. This is typically where Ghidra's translation of machine-readable binary code into something intended for human eyes ends and where the fastidious manual work of interpretation and annotation begins.

The human reverse engineer meticulously analyzes the decompiled code by repeatedly comparing it to the original assembly listing. This ensures no errors from the decompilation process are overlooked. As the engineer examines the code, they add explanatory comments and assign descriptive names to variables and functions to improve readability.





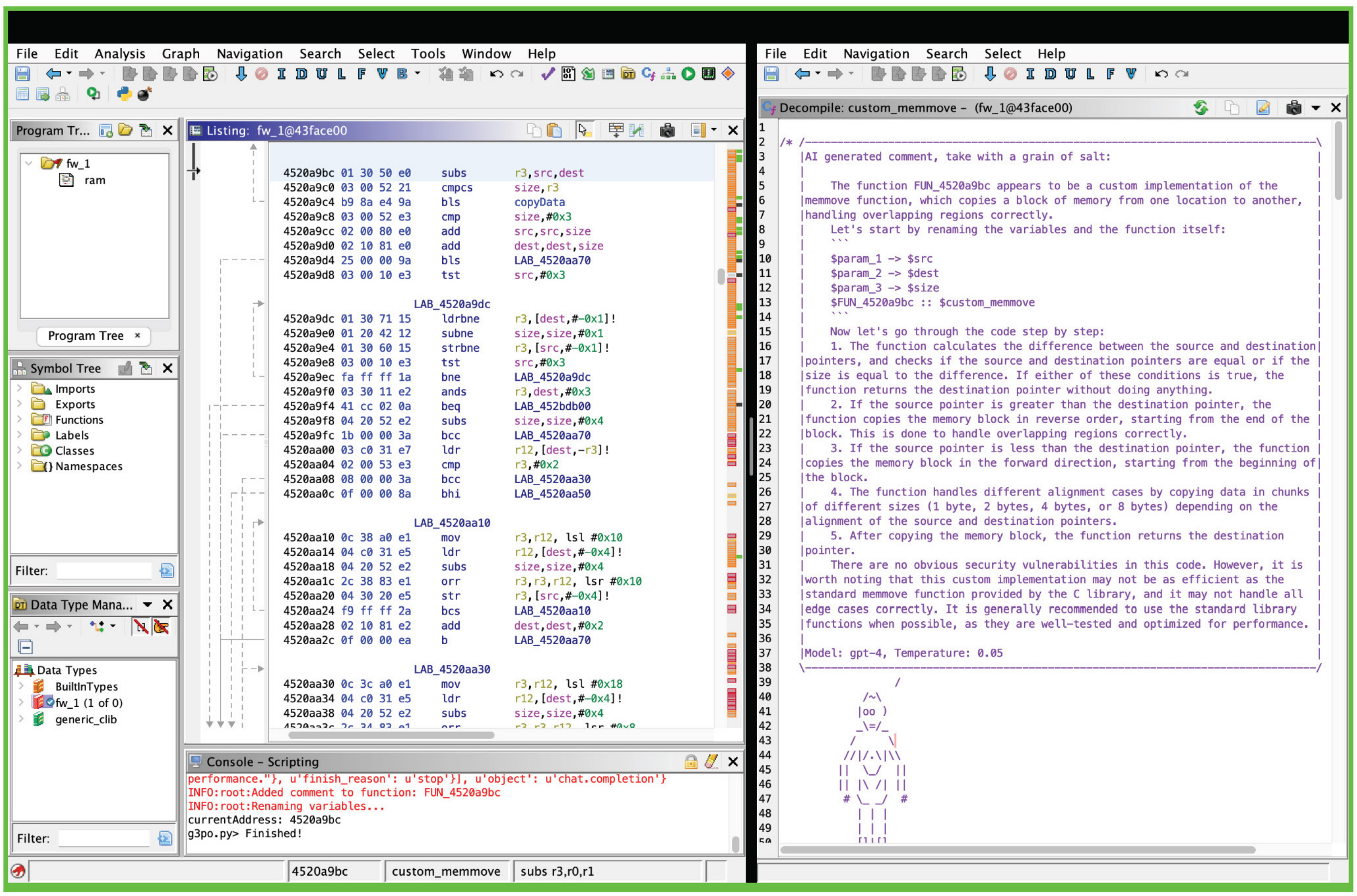




Tenable Research developed a tool, G-3PO, to pick up where Ghidra's decompiler leaves off, adding another layer of automation to the reverse engineering workflow. G-3PO submits a function's decompiled C code to a language model (it currently supports models from both OpenAI and Anthropic) and requests an explanation of what the function does along with suggestions for descriptive variable names. G-3PO can then automatically add these names and comments to the Ghidra decompilation listing.

This allows the reverse engineer to gain a rapid, high-level understanding of the code's functionality without having to first decipher every line. This bird's eye view of the binary in question provides the engineer with the information they need to direct their attention to the regions of code that most concern them, where they can resume the meticulous task of manual binary analysis.

Below, we see G-3PO commenting on an optimized memory-copying function pulled from the firmware binary of a Canon printer and decompiled by Ghidra. Interested readers can peruse the decompiled C code, both before G-3PO added its annotations and after, using either OpenAI's GPT-4 (shown below) or Anthropic's Claude v1.2 as its backend language model, respectively.



```
Unset
AI generated comment, take with a grain of salt:

The function FUN_4520a9bc appears to be a custom implementation of the
memmove function, which copies a block of memory from one location to
another, handling overlapping regions correctly.

Let's start by renaming the variables and the function itself:

Sparam 1 -> $src $param_2 -> $dest

Sparam_3 -> $size

$FUN_4520a9bc :: $custom_memmove

Now let's go through the code step by step:

The function calculates the difference between the source and
destination pointers. and checks if the source and destination pointers
are equal or it the size is equal to the difference. If either of these
conditions is true, the function returns the destination pointer without
doing anything.

If the source pointer is greater than the destination pointer, the
function copies the memory block in reverse order, starting from the end
of the block. This is done to handle overlapping regions correctly.

If the source pointer is less than the destination pointer, the function
copies the memory block in the forward direction, starting from the
beginning of the block.

4. The function handles different alignment cases by copying data in
chunks of different sizes (1 byte, 2 bytes, 4 bytes, or 8 bytes)
depending on the alignment of the source and destination pointers.

5. After copying the memory block, the function returns the destination
pointer.

There are no obvious security vulnerabilities in this code. However, it
is worth noting that this custom implementation may not be as efficient
as the Istandard memmove function provided by the C library, and it may
not handle all edge cases correctly. It is generally recommended to use
the standard library functions when possible, as they are well-tested
and optimized for performance.

Model: gpt-4, Temperature: 0.05
```

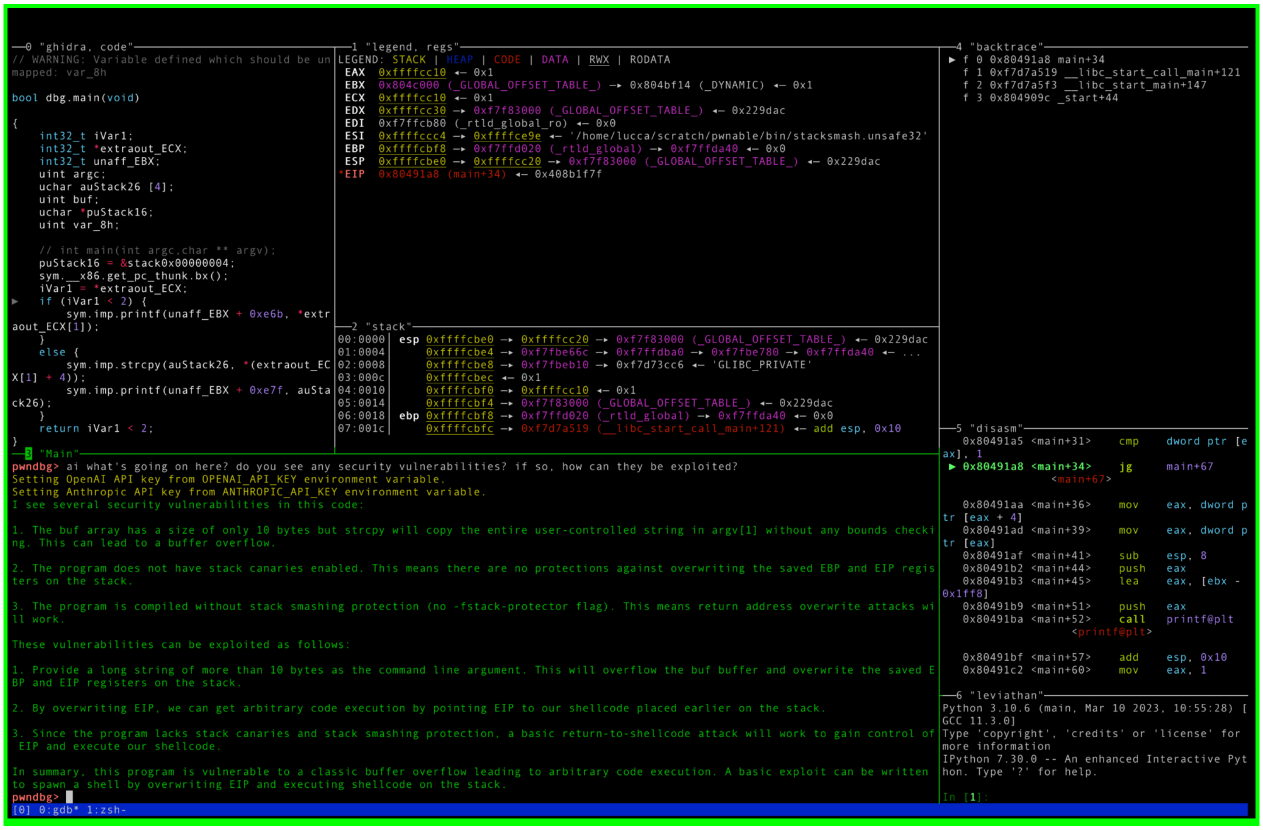
G-3PO isn't foolproof, and a good reverse engineer always maintains a healthy distrust of their tools. But the advantage of this particular application domain is that the model's suggestions can easily be checked against both the decompiler output and the assembly listing from which that output was derived.

An AI assistant for GDB with Pwndbg or GEF

Debugging code requires understanding intricate technical details across multiple areas. It's a discipline of reverse engineering that takes considerable effort to master. Naturally, this is a great opportunity to use AI to reduce complexity. To demonstrate this, Tenable Research developed an AI assistant for the GNU Debugger (GDB) to simplify the debugging process. Our tool was implemented as a plugin for two of the most popular GDB extension frameworks: GEF and Pwndbg. The tool supports language models from Anthropic and OpenAI, allowing it to analyze debugging information and answer questions about runtime state or assembly code.

Our GDB assistant reduces the complexities of debugging by providing an interactive tool for exploring the debugging context. It receives information on registers, stack values, backtrace, assembly and decompiled code if using the Ghidra extension in Pwndbg, providing as much of the relevant context as possible to accompany the user's queries. The user can pose whatever question they like to the model — from general queries like "What's going on here?" or "Does this function look vulnerable?" to more specific questions like "Are there any circumstances that will lead to this function calling free() twice on the same pointer?" The user can then ask the model follow-up questions for the sake of clarification or correction.

Here, we see our AI assistant tool being used with the Pwndbg and splitmind extensions to GDB, backed by Anthropic's large language model, claude-v1.2. "What's going on here?" the user asks the assistant, "Do you see any security vulnerabilities? If so, how can they be exploited?"



And the AI responds:

```
Unset
I see several security vulnerabilities in this code:

1. The buf array has a size of only 10 bytes but strcpy will copy the
entire user-controlled string in argv[1] without any bounds checking.
This can lead to a buffer overflow.

2. The program does not have stack canaries enabled. This means there
are no protections against overwriting the saved EBP and EIP registers
on the stack.

3. The program is compiled without stack smashing protection (no
-fstack-protector flag). This means return address overwrite attacks
will work.

These vulnerabilities can be exploited as follows:

1. Provide a long string of more than 10 bytes as the command line
argument. This will overflow the buf buffer and overwrite the saved EBP
and EIP registers on the stack.

2. By overwriting EIP, we can get arbitrary code execution by pointing
EIP to our shellcode placed earlier on the stack.
```

```
3. Since the program lacks stack canaries and stack smashing protection,
a basic return-to-shellcode attack will work to gain control of EIP and
execute our shellcode.

In summary, this program is vulnerable to a classic buffer overflow
leading to arbitrary code execution. A basic exploit can be written to
spawn a shell by overwriting EIP and executing shellcode on the stack.
```

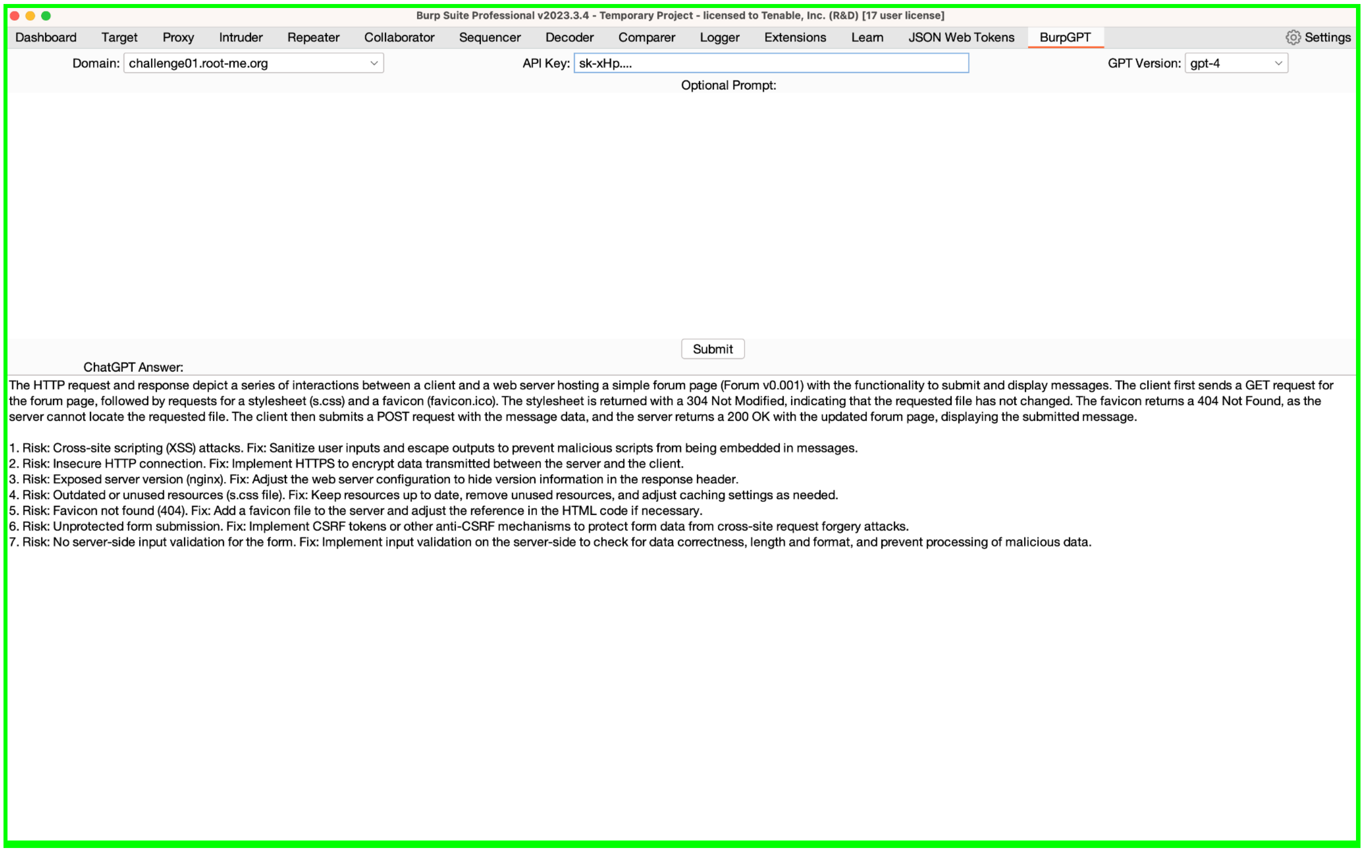
Our GDB assistant aims to solve the problem of navigating debugging's steep learning curve. It turns GDB into a conversational interface where users can discuss what the code is doing at each step instead of deciphering raw debugging data. The model guides users in analyzing vulnerabilities or other runtime behaviors in the target program.

Though not flawless, our GDB assistant, which has already been merged into pwndbg's upstream repository, shows promising results in reducing debugging complexity and time. We believe that it will not only prove to be a useful tool for professional reverse engineers, but a useful educational resource to help novices new to debugging and security research learn the ropes of this complex domain.

BurpGPT: an AI assistant for web app security research

Web applications provide a unique challenge for researchers due to the complexities in identifying vulnerabilities within them. Tools like our Tenable Web Application Scanning product provide comprehensive and automated vulnerability scanning for modern web applications. However, as researchers we continue to explore new avenues for exploitation. This research helps us to increase the efficacy of our products. One of the most powerful tools we utilize in our web application research is Burp Suite. Burp provides a fantastic suite of features to aid in manual testing for web app security vulnerabilities. In our efforts to identify ways in which AI could benefit web application security, we looked at how we could integrate ChatGPT and Burp Suite to identify common web application vulnerabilities and recommend solutions to address them.

We built BurpGPT as an extension for Burp Suite to allow researchers to utilize GPT for analysis of HTTP requests and responses. The tool works by leveraging Burp's proxy feature to intercept HTTP traffic and prompts the OpenAI API to analyze the traffic to identify risks and potential fixes to any issues identified. This can be used to discover injection points, misconfigurations and more. As we developed this extension, we tested some real-world scenarios and we found that GPT3.5 and GPT4 successfully identified Cross-Site Scripting (XSS) vulnerabilities and misconfigured HTTP headers without requiring any additional fine-tuning. These capabilities can have dramatic effects on reducing manual testing and automating security testing for web application developers. In addition, it gives our researchers another tool in their arsenal to help identify novel new exploitation techniques that can be implemented into our products.



EscalateGPT: An AI powered tool to identify IAM policy issues

When it comes to cloud security, misconfigurations in identity and access management (IAM) is one of the most common concerns for organizations and far too often overlooked. In fact, in the Tenable 2022 Threat Landscape Report, our research team found over 800 million exposed records attributed to cloud misconfigurations. Because IAM policy misconfigurations are so common, we set out to develop a tool to help identify IAM policy issues that could be incorporated and further enhance Tenable Cloud Security (formerly Tenable.cs). Enter EscalateGPT, a Python tool designed to identify privilege-escalation opportunities in Amazon Web Services (AWS) IAM.

This tool can be used to retrieve all IAM policies associated with users or groups and will then prompt the OpenAI API, asking it to identify potential escalation opportunities and any relevant mitigations. EscalateGPT returns results in a JSON format that includes the path, the Amazon Resource Name (ARN) of the policy that could be exploited for privilege escalation and the recommended mitigation strategies to address the identified vulnerabilities. In our testing against real-world AWS environments, we found that GPT4 managed to identify complex scenarios of privilege escalation based on non-trivial policies through multi-IAM accounts. As a comparison, using GPT3.5-turbo, we found that only half of the privilege escalation cases we tested for were identified.

```
python3 escalate_gpt.py -openai_key sk-x[redacted]
   # #   #  #  # #     #     ####  #    #   ##   ##### #     # #     #    #
  #   #  #  #  # #          #    # #    #  #  #    #   #       #     #    #
 #     # #  #  #  #####     #      ###### #    #   #   #  #### ######     #
 ####### #  #  #       #    #      #    # ######   #   #     # #          #
 #     # #  #  # #     #    #    # #    # #    #   #   #     # #          #
 #     #  ## ##   #####      ####  #    # #    #   #    #####  #          #

Starting AWS privilege escalation tool using ChatGPT
Get all policy from account 832970976958
Found 10 policies in use. Prompting ChatGPT for analysis
[
    {
        "path": "aviv-*->*",
        "policy": "*",
        "risk": "aviv has full access to all resources, including others' user accounts",
        "mitigation": "Limit aviv's permissions to only necessary actions and resources"
    },
    {
        "path": "naum-*->*",
        "policy": "*",
        "risk": "naum has full access to all resources, including others' user accounts",
        "mitigation": "Limit naum's permissions to only necessary actions and resources"
    },
    {
        "path": "oksana-*->*",
        "policy": "*",
        "risk": "oksana has full access to all resources, including others' user accounts",
        "mitigation": "Limit oksana's permissions to only necessary actions and resources"
    },
    {
        "path": "test_user-AdministratorAccess->*",
        "policy": "AdministratorAccess",
        "risk": "test_user has full access to all resources, including others' user accounts",
        "mitigation": "Limit test_user's permissions to only necessary actions and resources"
    }
]
```

CONCLUSION

As LLMs evolve and are trained with more specific data to suit targeted use cases, it's to be anticipated that malicious actors will take advantage of these emerging technologies for exploit development and other nefarious uses. We expect to see advancements in more convincing and linguistically accurate phishing emails. Attackers will look to abuse public models to provide tainted data or trick the models into disclosing sensitive data provided to the models unknowingly by everyday users. In fact, organizations across the globe are already warning their staff not to supply these LLM services with internal or sensitive data, some even going as far as blocking access to these tools on corporate networks. While LLMs are producing surprisingly accurate code snippets, they are not yet rising to the level of experienced malware developers. However with the breakneck speed in advancements of these models, it's only a matter of time before the threat of reliable AI-generated malware is realized.

The silver lining is that there's ample opportunity for defenders to harness this technology as well. From log parsing and anomaly detection to triage and incident response capabilities, defenders could have the upper hand. In addition to our examples of using LLMs like ChatGPT to reduce the manual workload of reverse engineering tasks and security research, another avenue where AI could prove to be a key tool for development teams is static code analysis to identify potentially-exploitable code. Coupled with advanced threat detection and intelligence from trained AI models, there is an abundance of use cases to aid defenders. While we're only at the start of our journey in implementing AI into tools for security research, it's clear the unique capabilities these LLMs provide will continue to have profound impacts for both attackers and defenders.

About Tenable

Tenable® is the Exposure Management company. More than 40,000 organizations around the globe rely on Tenable to understand and reduce cyber risk. As the creator of Nessus®, Tenable extended its expertise in vulnerabilities to deliver the world's first platform to see and secure any digital asset on any computing platform. Tenable customers include approximately 60 percent of the Fortune 500, approximately 40 percent of the Global 2000, and large government agencies. Learn more at www.tenable.com.

ABOUT THE AUTHORS

Scott Caveza

Staff Research Engineer

Olivia Fraser

Staff Research Engineer

Yossi Nisani

Senior Software Engineer