Exploring Applicability of LLM-Powered Autonomous Agents to Solve Real-Life Problems: Microsoft Entra ID Administration Agent (MEAN)

Roberto Rodriguez\(^1\) and Nestori Syynimaa\(^1\)\(^2\)\(^a\)\(^b\)

\(^1\)Microsoft Security Research (MSecR), U.S.A.
\(^2\)Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland

Keywords: LLM, Autonomous Agent, ChatGPT, Powershell, Entra ID.

Abstract: Microsoft Entra ID is Microsoft’s identity and access management solution used by many public and private sector organisations globally. In March 2023, Microsoft retired two PowerShell modules which have enabled automation of administrative tasks, such as user management. The replacement module is based on Microsoft Graph API, and its effective usage would require administrators to learn software development skills. In this paper, we will report the results of work-in-progress research on exploring the applicability of LLM-powered autonomous agents to solve real-life problems. We describe the design and proof-of-concept implementation of MEAN, an agent that performs Entra ID administrative tasks using Microsoft Graph API based on natural language prompts. The results show that LLM-powered autonomous agents can perform at least simple Entra ID administrative tasks. This indicates that the agents could ease the administrative burden by removing the need to learn software development skills.

1 INTRODUCTION

1.1 Microsoft Entra ID

Microsoft Entra ID (MEID) is Microsoft’s cloud-based identity and access management (IAM) solution (Microsoft, 2023c), formerly known as Azure Active Directory (Azure AD). MEID is widely used globally in public and private sector organisations (see Syynimaa, 2022). Administrators can administer Entra ID via web-based interfaces like the Entra ID portal. The portals are suitable for changing Entra ID-wide settings and performing manual tasks involving a small number of individual targets, such as users or groups. However, portals do not scale, which forces administrators to rely on automation solutions. For instance, administrators may need to quickly disable thousands of users during a cyber security attack to prevent further damage. This is not possible using the Entra ID portal.

PowerShell is a task automation solution consisting of a scripting language and a command-line shell (Microsoft, 2023d). Microsoft has provided PowerShell modules for administering Entra ID. The first module, MSOnline, was published in the early 2010s, and the next one, AzureAD, was published later in the 2010s. Both were built to help administrators perform their duties, like managing Entra ID users and licenses. However, these modules are now set to be deprecated in March 2024 (Bash, 2023). The reason for this is that the underlying Application Programming Interfaces (APIs) the modules are using are deprecated at the same time. The only supported API after that is Microsoft Graph (MSGraph) API (Microsoft, 2023b).

Microsoft offers a replacement module for administrators called “Microsoft Graph PowerShell SDK” (Microsoft, 2023a). The module is an API wrapper for the MSGraph API, created automatically from the MSGraph API schema (Microsoft, 2023a). This is problematic for administrators as the module is not built to help them perform their tasks but allows...
them to call the MSGraph API directly from PowerShell. Even the name of the module contains the acronym SDK, which refers to Software Development Kit. According to the global skills and competency framework for the digital world (SFIA), software development skills are not part of the system administrator role skill set (SFIA, 2023). Mastering the new module would require learning software development, especially MSGraph API.

1.2 Large Language Model

The large Language Model (LLM) is a special case of the Pre-trained Language Model (PLM). The challenge with PLMs is that they are unable to perform unseen tasks without task-specific training (Kalyan, 2024). Generative Pre-trained Transformers (GPTs) are using Natural Language Processing (NLP) for task-agnostic pre-training (Brown et al., 2020). Still, task-specific examples or datasets are required on top of general training to perform the desired task (Brown et al., 2020).

The public and scientific interest in LLMs has significantly increased after OpenAI published ChatGPT (OpenAI, 2024) for the general public in November 2022. For instance, ChatGPT articles published in arXiv have risen from zero in December 2022 to over 300 by May 2023 (Liu et al., 2023).

ChatGPT has been used successfully to generate software code (Liu et al., 2023) and to teach software development skills (Hellas et al., 2023). However, there have also been some challenges. For instance, the quality of the generated code cannot be guaranteed (Liu et al., 2023), and the judgements made may not be grounded in reality (Faggioli et al., 2023). This is often referred to as hallucination (Händler, 2023). Moreover, as an LLM, ChatGPT still lacks domain-specific knowledge (Faggioli et al., 2023).

1.3 Autonomous Agents

LLMs have been found to struggle with maintaining consistent logic across chains of reasoning (Händler, 2023). Autonomous LLM-powered agents can break user-prompted goals into smaller tasks (Händler, 2023). These tasks can be assigned to specialised agents utilising contextual resources such as tools (Händler, 2023). The balance between the autonomy of LLM-powered agents and the alignment of human users is illustrated in Figure 1. In other words, the agent should be self-organising if the user needs to perform tasks that demand active participation (L2).
1.4 Research Questions

Our research sought to answer the following research question:

How can we implement an autonomous LLM-powered agent to help administrators perform Entra ID administrative tasks?

1.5 Structure of the Paper

The rest of the paper is structured as follows. The building process of the Microsoft Entra ID Administration agent (MEAN) is described in the next section. The last section, discussion, concludes the paper.

2 BUILDING THE MEAN

This paper reports a Design Science Research (DSR) study (see vom Brocke, Hevner, & Maedche, 2020), and we’ll follow the DSR process guidelines by Peffers et al. (2007) in this section. Following Onggo (2012), the resulting agent is illustrated using BPMN 2.0 notation.

2.1 Motivation

The problem and motivation for the research were explained in the introduction section. To sum up, Microsoft is deprecating existing Entra ID PowerShell modules, and the replacement module requires learning new skills that are not part of administrators’ skillset.

2.2 Objectives

The solution’s objective is to build an agent that helps administrators perform their tasks without needing to learn software development skills.

2.3 Design and Development – Round 1

Jupyter Notebooks (Jupyter, 2024) was selected as a development platform for the agent because it supports Python, giving easy access to ChatGPT-4 API. ChatGPT-4 supports function calling, which allows agents to call external APIs (OpenAI, 2023). The available functions and parameters can be sent to ChatGPT with the user prompt. The functions and parameters should be in the format that ChatGPT can interpret. The MSGraph API documentation (see Microsoft, 2024a) is not structured enough for ChatGPT. However, ChatGPT supports OpenAPI specification (see OpenAPI, 2021). Microsoft publishes MSGraph API documentation in OpenAPI format (see Microsoft, 2024a), allowing it to be used with ChatGPT as-is.

The agent design is illustrated in Figure 3. The user sends the OpenAPI specification along with a prompt to the agent. Agent interprets (reasons) the input and plans the next steps based on the user prompt and the given OpenAPI specification. The agent starts to execute the plan by calling MSGraph API. The response from the API call is returned to the reasoning and planning process. When the plan is completed, the response is returned to the user.

The source code of the proof-of-concept (PoC) agent that implements the design is available on GitHub (see Rodriguez, 2023a). The PoC uses the MSGraph API users endpoint, which provides functionality for searching, creating, updating, and deleting users.

2.4 Demonstration – Round 1

The output of running the PoC agent is also available on GitHub (see Rodriguez, 2023a). The output shows the result of a simple query to find a username based on a given object ID.

After receiving the input from the user, the agent starts building a plan. First, it makes an observation of which API it could use to get the requested information. This leads to a thought that the agent is ready to execute the API call. Next, the agent starts acting on the plan and executes the request_get tool to make the actual API call. From the response, the agent makes an observation that the name of the user is “Example User”. This leads to a thought that user information is successfully retrieved and the plan is successfully executed. The agent makes a final observation that the name of the user whose id was given is “Example User”. This leads to a thought that the plan is finished and the information the user asked for is retrieved.

However, the agent was not able to perform more complex tasks. When asked to list all users whose password has been reset over six months ago, the agent answered I’m sorry, but this API does not provide the functionality to filter users based on the time their password was reset. The API returns only a subset of properties for users by default (see Table 1). All other attributes, such as lastPasswordChangeDateTime, would require using the $select query parameter. To us, it seemed that the agent was only aware of the common properties, not all available properties.
Table 1: Common properties of users (Microsoft, 2024b).

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The unique identifier for the user.</td>
</tr>
<tr>
<td>businessPhones</td>
<td>The user's phone numbers.</td>
</tr>
<tr>
<td>displayName</td>
<td>The name displayed in the address book for the user.</td>
</tr>
<tr>
<td>givenName</td>
<td>The first name of the user.</td>
</tr>
<tr>
<td>jobTitle</td>
<td>The user's job title.</td>
</tr>
<tr>
<td>mail</td>
<td>The user's email address.</td>
</tr>
<tr>
<td>mobilePhone</td>
<td>The user's cellphone number.</td>
</tr>
<tr>
<td>officeLocation</td>
<td>The user's physical office location.</td>
</tr>
<tr>
<td>preferredLanguage</td>
<td>The user's language of preference.</td>
</tr>
<tr>
<td>surname</td>
<td>The last name of the user.</td>
</tr>
<tr>
<td>userPrincipalName</td>
<td>The user's principal name.</td>
</tr>
</tbody>
</table>

Moreover, the agent seemed to be capable of showing only the 16 first users (sorted by userPrincipalName property). However, the API returns by default 100 users and allows returning up to 1000 users per request using $top query parameter. Again, it seemed that the agent was not aware of this parameter.

2.5 Design and Development – Round 2

After the first design and development round, the agent seemed to work only partly as intended. It was able to perform simple tasks but didn’t seem to understand the API correctly.

The original MS Graph API Open API specification YAML file had 28021 rows in total. Validating the file with the OpenAPI online validator (see APIDevTools, 2022) crashed the browser. The first lines of the file (see Figure 4) show that the file

Figure 3: MEAN Design.
is using a lot of internal $ref clauses. Changing the setting of the online validator to skip circular references threw an error message “Circular $ref pointer found”. For the same reason, in the first development round, we had to use dereference=False option when calling reduce_openapi_spec function (see Rodriguez, 2023b). That effectively did not follow any $ref clauses and thus removed crucial information about the API. Moreover, reduce_openapi_spec ignored all query parameters that were not marked as required (see Langchain, 2023, step 3).

To fix the aforementioned issues, the MS Graph Open API specification was manually modified to remove unnecessary circular references and to change the type of all query parameters to required (see Syynimaa, 2024).

2.6 Demonstration – Round 2

The output of running the PoC agent to list lastPasswordChangeDateTime and jobTitle properties of users is available on GitHub (see Rodriguez & Syynimaa, 2024b). The output shows that the agent now understands the API better, i.e., how to use $select query parameter. This time, the API threw an error indicating that $select query parameter was specified more than once, which the agent observed correctly. The resulting thought was that it should use just one $select parameter and comma to separate the values. Next observation was the returned user information and the resulting thought that the data was successfully retrieved, and it could now be returned.

The process seems the same as you would expect from a natural person to execute a similar task. Instead of a natural person, the user could ask the autonomous LLM-powered agent to retrieve the needed information.

2.7 Evaluation

The conventional way to perform administrative tasks, such as finding a user with the given ID, would be to use an MSGraph PowerShell module. This would require the administrator using the module to know which API calls and corresponding PowerShell cmdlets they should use.

The designed agent allowed a user to complete a task that used MSGraph API without a need to learn software development skills. As such, the agent fulfils the set objectives. However, due to limitations of the current LangChain implementation, especially the reduce_openapi_spec function, do not allow the agent to release the full potential of ChatGPT. Even the source code of the function says “This is a quick and dirty representation of OpenAPI specs” (Langchain, 2023).

Figure 4: Excerpt of OpenAPI-MSGraph-v1.0-Users.yml file.
2.8 Communication

The process and output of building the agent are reported in this research paper. The source code and Jupyter notebook are published on GitHub (see Rodriguez & Syynimaa, 2024a) and are freely available for anybody to download.

3 DISCUSSION

3.1 Implications to Practice

We demonstrated that an LLM-powered autonomous agent can help administrators perform tasks that would otherwise require software development skills. This allows administrators to focus on daily activities instead of learning software development skills. Moreover, this greatly improves administrators’ efficiency in performing administrative tasks, especially in stressful situations like during cyber-attacks. However, due to limitations in the components used in the MEAN implementation, it is not mature enough to perform the day-to-day tasks expected from administrators.

3.2 Implications to Science

In this paper, we designed and implemented a PoC of an autonomous agent capable of completing tasks by invoking MSGraph API based on the user’s natural language input. This proves that, despite the abovementioned challenges, LLM-powered autonomous agents can perform simple Entra ID administrative tasks. As such, the study supports the findings of previous studies (Nascimento, Alencar, & Cowan, 2023; Topsakal & Akinci, 2023).

3.3 Limitations

This study focused on performing tasks using a specific external tool, MSGraph API. Other APIs may yield different results.

3.4 Directions for Future Research

An interesting avenue for future research would be to study could agents be used to call PowerShell commands instead of back-end APIs. PowerShell module metadata, e.g., function descriptions and parameter details, can be extracted programmatically. This could lead to a more generalised solution which is not limited to cloud services and could also allow using agents to perform tasks on local computers.

REFERENCES


Liu, Yiheng, Han, Tianle, Ma, Siyuan, Zhang, Jiayue, Yang, Yuanyuan, Tian, Jiaming, Ge, Bao. (2023). Summary of ChatGPT-Related research and perspective towards the future of large language models. Meta-Radiology, 1(2), 100017. doi:10.1016/j.metrad.2023.100017


OpenAI. (2024). Introduction to ChatGPT. Retrieved from https://openai.com/blog/chatgpt


Rodriguez, Roberto, & Syynimaa, Nestori. (2024b). Microsoft Entra ID Administration Agent (MEAN). Retrieved from https://github.com/OTRF/MEAN/blob/ee7c4e21462d9c5f4dad974774c5499e5b0c5d670(notebooks/openapi-specs/OAI-MSGraph-Users-Agent.ipynb


Exploring Applicability of LLM-Powered Autonomous Agents to Solve Real-life Problems: Microsoft Entra ID Administration Agent (MEAN)