Using Large Language Models to Extract Planning Knowledge from Common Vulnerabilities and Exposures

Tim Oates[1], **Ron Alford**[2], **Shawn Johnson**[2], **Cory Hall**[3]

[1] University of Maryland, Baltimore County
[2] The MITRE Corporation
[3] Cyber Domain Ontology, Linux Foundation
oates@umbc.edu, ralford@mitre.org, spjohnson@mitre.org, cory.hall@cyberdomainontology.org

Abstract

Understanding attackers' goals and plans is crucial for cyber defense, which relies on understanding the basic steps that attackers can take to exploit vulnerabilities. There is a wealth of knowledge about vulnerabilities in text, such as Common Vulnerabilities and Exposures (CVEs), that is accessible to humans but not machines. This paper presents a system, called CLLaMP, that uses large language models (LLMs) to extract declarative representations of CVEs as planning operators represented using the Planning Domain Description Language (PDDL). CLLaMP ingests CVEs, stores them in a database, uses an LLM to extract a PDDL action that specifies preconditions for, and the effects of, the exploit, and updates the database with the action. The resulting planning operators can be used for automatically recognizing attacker plans in real time. We propose metrics for evaluating the quality of extracted operators and show the translation results for a set of randomly selected CVEs.

Introduction

Some cyber-attacks are mostly automated, like denial-of-service attacks in which large numbers of machines run software that remotely consumes resources of target systems, denying access by legitimate users. Others, like advanced persistent threats, involve humans executing attack plans with many phases and steps over long periods of time (Korban et al. 2017). Successful defense against such attacks requires recognizing activity as part of an attack plan, understanding what next steps may further the attackers' goals, and intervening to prevent those steps (Hoffmann 2015). Said differently, plan recognition is a key element of what humans do when defending cyber systems. But recognizing attackers' plans requires deep domain knowledge, about the system under attack, possible vulnerabilities, and possible sequences of actions in attack plans (Kouremetis et al. 2024).

In this paper we explore the possibility of automatically extracting declarative representations of planning operators – specifications of the conditions under which an attacker's actions can have an intended effect – from text describing known vulnerabilities. Plans are sequences of actions (planning operators) such that when the preconditions of one action are met, the attacker can change the state of the system under attack, thereby enabling the next action in the plan to succeed. Planning operators are the raw materials needed to drive planning and plan recognition in any domain.

In particular, we focus on the vast amount of cybersecurity knowledge captured in common vulnerabilities and exposures (CVEs) and using large language models (LLMs) to represent each CVE as a planning operator in the planning domain definition language (PDDL) (Fox and Long 2003). Consider the following description from CVE-2023-2387:

> A vulnerability classified as problematic was found in Netgear SRX5308 up to 4.3.5-3. Affected by this vulnerability is an unknown functionality of the file scgi-bin/platform.cgi?page=dmz_setup.htm of the component Web Management Interface. The manipulation of the argument winsServer1 leads to cross-site scripting. The attack can be launched remotely. The exploit has been disclosed to the public and may be used. The identifier VDB-227665 was assigned to this vulnerability. NOTE: The vendor was contacted early about this disclosure but did not respond in any way.

A human who wanted to use this knowledge to attack a system would check to see if it is using any version of the Netgear SRX5308 product, which is a 4-port SSL VPN firewall, up to and including version 4.3.5-3. They would then attempt to remotely access the system to manipulate the winServer1 argument of scgi-bin/platform.cgi?page=dmz_setup.htm with the goal of being able to perform cross-site scripting. We will talk more about the PDDL representation of planning knowledge in the next section, but an action in PDDL that captures the knowledge encoded in the CVE is shown below.

```
(:action EXPLOIT-CVE-2023-2387
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'Netgear SRX5308')
                     (has_version 'Netgear SRX5308' 'up to 4.3.5-3')
                     (has_argument ... ?a)
                     (manipulates ?a 'winsServer1'))
  :effect (and (gain-access ?a ?s 'cross site scripting')))
```

The action above indicates that there must be a system and an attacker, that the attacker must have remote access to the system, that the system has a vulnerable version of Netgear SRX5308, and that the attacker manipulates the winServer1 argument of the file scgi-bin/platform.cgi?page=dmz_setup.htm.[1] Doing so leads to the effect that the attacker gains the ability to perform cross-site scripting on the system. From the perspective of a defender, the operator above can be used in several ways. For example, the attack is only possible with certain versions of Netgear SRX5308. If later versions are present, the attack cannot proceed, and if a vulnerable version is present then an upgrade is in order. Further, frequent manipulation of the winServer1 argument of dmz_setup.htm is an indicator that the user doing so may be an attacker whose next step is to exploit cross-site scripting.

The contributions of this paper include (1) a system architecture for extracting structured planning operators from CVEs, (2) definitions of metrics for evaluating various aspects of the quality of the extracted operators, (3) insights about various ways of prompting LLMs for this use case and the final prompt approach (included in Appendix A), and (4) empirical results show the utility of the overall approach.

The remainder of this paper is organized as follows. The next section describes background on CVEs, PDDL, and LLMs, as well as related work in which LLMs are used in various ways for planning tasks. That is followed by a description of our system – called CLLaMP[2] – for extracting planning operators from CVEs as well as the data that it ingests. The next two sections outline how such a system can be evaluated and present an empirical analysis of the operators extracted by CLLaMP, respectively. The final section concludes and points to ample future work.

Background and Related Work

Because CLLaMP uses CVEs (the C in CLLaMP) and large language models (the LLaM) to extract planning operators in PDDL (the P), this section briefly reviews all three topics.

Common vulnerabilities and exposures (CVEs) are maintained by an international community of members in the CVE Program with the aim of cataloging publicly disclosed vulnerabilities in a consistent manner. At the time this paper was written there were more than 200K CVEs available for search and download at cve.org. Each CVE is represented in JSON format in a single file. The most important part of that JSON for the purposes of this paper is the textual description of the vulnerability, such as the one shown in the previous section. Those descriptions are free form text and, despite efforts at standardization, exhibit all the variation one expects to see in natural language descriptions of real-world phenomena like cybersecurity vulnerabilities. Figure 2 contains a histogram of the lengths of CVE descriptions in characters. Note that some are quite long (a few thousand characters) but most are short, with fewer than 500 characters.

The CVE Program exists to capture information about vulnerabilities so that cyber professionals can keep their systems safe by exposing information about actions that attackers may take to compromise those systems. Thus, the focus on CVEs in this work.

A crucial prerequisite to enabling automated recognition of attacker plans is creating declarative representations of actions that attackers may take in the cybersecurity domain. The Planning Domain Definition Language (PDDL) is a family of standards for describing planning tasks (Fox and Long 2003). It was initially developed in 1998 to enable a planning competition, where groups with various planners could ingest planning domain definitions in a common format and compare results.

Planning problems have common elements, such as objects in the domain of interest, predicates that specify properties of objects and relationships among objects, an initial state, a goal state, and a set of actions. Consider the following CVE (CVE-2017-4889):

> VMware Workstation Pro/Player 12.x before 12.5.3 contains a security vulnerability that exists in the SVGA driver. An attacker may exploit this issue to crash the VM or trigger an out-of-bound read. Note: This issue can be triggered only when the host has no graphics card or no graphics drivers are installed.

It mentions objects like software (“VMware Workstation Pro/Player” and “SVGA driver”), version numbers (“12.5.3”), and hardware (“graphics card”). It describes relationships between objects, such as VMware Workstation Pro/Player having a version before 12.5.3, and a host not having a graphics card. If the system is in an initial state of having a vulnerable version of the software without a graphics card, then an attacker can take the action of exploiting the SVGA driver to achieve the goal of crashing the VM. This is all captured in the planning operator below (which was extracted by an LLM) in PDDL format.

```
(:action EXPLOIT-CVE-2017-4899
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_component ?s 'VMware Workstation Pro/Player')
                     (has_version ?s '12.x')
                     (has_no_graphics_card ?s)
                     (has_no_graphics_drivers ?s))
  :effect (and (crash-vm ?s ?a)
               (trigger-out-of-bound-read ?s ?a)))
```

The action name associates the PDDL operator with the exploitation of the vulnerability described in a particular CVE. It takes two parameters denoted by the variables ?s and ?a. The preconditions assert that the former must be a system and the latter an attacker, that the system have a vulnerable version of the software, and that the system has no graphics card and no graphics drivers. That is, the preconditions ensure that the system under attack is vulnerable in the way described by the CVE. The effects of the attacker’s action, their short-term goal, is to crash the VM and/or trigger an out-of-bound read.

Given a collection of planning operators in this format, a standard planner, and a corresponding PDDL problem file that describes the objects in, and initial state of, compute infrastructure, one can either plan attacks or recognize attacker plans.

[1] For readability, we use strings instead of object literals. The file name was also omitted from the planning operator for the same reason.

[2] Code available at https://github.com/ronwalf/CLLaMP

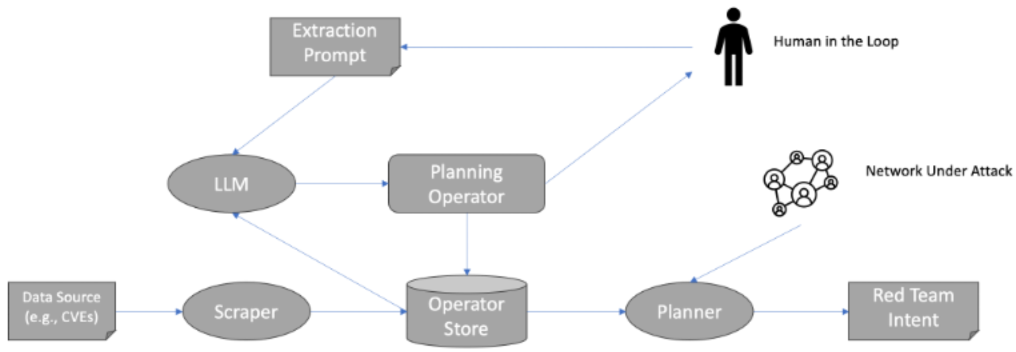

Figure 1: CLLaMP system architecture

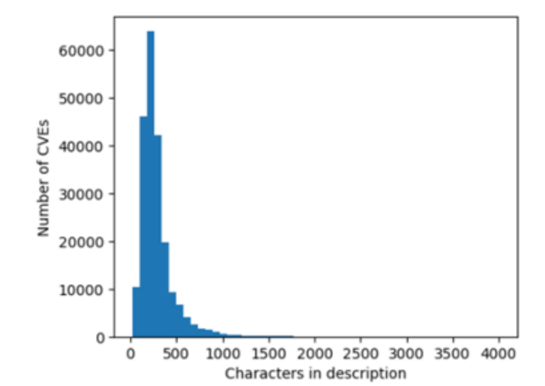

Figure 2: Histogram of CVE description lengths in characters.

A difficult part of making this work is turning CVE descriptions into structured PDDL representations. To that end we are exploiting large language models (LLMs) which have shown to be effective creating PDDL (Guan et al. 2024; Oswald et al. 2024), and other tasks such as turning natural language queries into SQL queries (Li et al. 2024) and turning natural language descriptions of functionality into program code (Jin et al. 2023). That is possible because the corpora used to train the LLMs contain large amounts of source code (e.g., from GitHub™) and examples of SQL queries (e.g., from web pages documenting SQL or posts on Stack Exchange™answering questions about how to write queries). Both interest in PDDL and examples of PDDL are significantly more sparse, so there is far less work with LLMs and PDDL, and it was not clear a priori how much LLMs know about the structure of PDDL. Of note, in the previous example ChatGPT is willing to assume the existence of string literals in PDDL, a common language feature which is not a part of the PDDL standards.

System and Data

The architecture of CLLaMP is shown in Figure 1. Data in the form of CVEs is ingested into an operator store. The CVEs used in this paper were downloaded from https://www.cve.org/ on September 7, 2023 in CVE JSON 5.0 format. They are then stored in the NoSQL database MongoDB, which is a good match for the JSON format of the underlying CVEs.

When new CVEs are added to the collection, an LLM is triggered to extract planning operators, one per CVE, as done in similar text-to-PDDL approaches (Guan et al. 2024; Oswald et al. 2024). We used OpenAI's gpt-3.5-turbo model for this work. The LLM uses a hand-crafted prompt to drive the extraction process. The PDDL content expressing the planning operator is parsed from the LLM's response to the prompt, which is then added to the document containing the original CVE. Special purpose code was written to manage OpenAI API timeouts, which can be frequent depending on factors such as the number of concurrent users and OpenAI's internal resources devoted to handling API requests.

Documents in the MongoDB collection have the form shown in Figure 3. Each document corresponds to a CVE and has the unique CVE ID, the date it was published, and the date that it was ingested into the system database. The entire raw JSON file is stored in the database so that it is available for fast reprocessing as the system's code changes. The text that describes the vulnerability is stored in its own field for ease of access. Another field is created when the CVE's description is converted to PDDL, which contains the corresponding planning operator as a formatted text string and the date the operator was extracted.

A significant amount of work went into designing the prompt used by the LLM. The simple approach of "Below is a CVE, please turn it into an action in PDDL" did not work. The LLM clearly had knowledge of CVEs and PDDL, but the prompt was not specific enough. The LLM created a wide array of predicates, often ignored crucial information in the CVE, and tended to create actions for reporting the CVE instead of capturing how it could be exploited. Our next attempt included chain-of-thought prompting (Wei

```
{
  cve_id: ID assigned to the CVE, e.g., CVE-2022-0991
  raw_data: The raw contents of the file containing the CVE as a string
  date_published: The date the CVE was originally published
  date_inserted: The date the CVE was inserted into the database
  description: The text of the description of the CVE

  pddl {
    operator: The planning operator extracted from the CVE
    date_inserted: The date the operator was inserted into the database
  }
}
```

Figure 3: Structure of documents in the MongoDB collection

et al. 2023) to get the LLM focused on the right information, such as asking for a table of system components and version numbers before asking for the planning operator. Though the LLM produced the table, it would still not include that information in the operator. Requests specifically asking for the table contents to appear in the operator's preconditions were ignored. LLMs are known to have problems with instructability.

The approach that finally worked was in-context learning (Dong et al. 2022), i.e., giving several examples of CVE descriptions and the desired PDDL output. One version of such a prompt is shown in Appendix A. The advantage of this few-shot learning approach is that it is easy to address extraction errors (see the architecture diagram) by having humans review planning operators and updating the prompt with additional examples.

Finally, given the store of planning operators and a PDDL problem file built from an existing cyber-system, such as a computer network, that describes the existing objects and their properties/relations, it is possible to reason automatically about red team (attacker) goals, plans, and intents.

4. Experiment Plan

This section describes how we evaluated the output of the LLM, which is by constructing gold standard actions for a set of CVEs not included in the LLM's prompt. Actions in PDDL refer to objects and predicates in a logical structure. These elements were evaluated using the following metrics.

- **Object recall:** Let O^G be the set of domain objects mentioned in the gold standard operator. Let O^L be the set of domain objects mentioned in the operator extracted by the LLM. Object recall is defined as $\frac{|O^G \cap O^L|}{|O^G|}$. That is the fraction of objects mentioned in the gold standard that are also mentioned in the extracted operator. Note that object mentions occur as arguments to predicates.
- **Object precision:** The fraction of domain objects mentioned in the extracted operator that are also mentioned in the gold standard is $\frac{|O^G \cap O^L|}{|O^L|}$.
- **Predicate recall**: Let P^G be the set of partially instantiated predicates (predicate name and arity, ignoring argument values) in the gold standard operator. Let P^L be the set of partially instantiated predicates mentioned in the operator extracted by the LLM. Predicate recall is defined as $\frac{|P^G \cap P^L|}{|P^G|}$.
- **Predicate precision**: Similarly, predicate precision is $\frac{|P^G \cap P^L|}{|P^L|}$.
- **Logic edit distance**: To measure the quality of the structure of the planning operator we compute the edit distance between the gold standard structure and the extracted structure. This metric ignores predicate names and arguments, and simply seeks to ensure that the predicates are present in the correct logical form. The example below should clarify how this is computed.

Consider the following gold standard action for the extracted operator shown earlier:

```
(:action EXPLOIT-CVE-2017-4899
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_component ?s 'VMware Workstation Pro/Player')
                     (has_version ?s 'before 12.5.3')
                     (manipulates ?a 'SVGA driver')
                     (or (has_no_graphics_card ?s)
                      (has_no_graphics_drivers ?s)))
  :effect (and (crash-vm ?s ?a)
          (trigger-out-of-bound-read ?s ?a)))
```

The extracted operator is shown here for convenience:

```
(:action EXPLOIT-CVE-2017-4899
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_component ?s 'VMware Workstation Pro/Player')
                     (has_version ?s '12.x')
                     (has_no_graphics_card ?s)
                     (has_no_graphics_drivers ?s))
  :effect (and (crash-vm ?s ?a)
          (trigger-out-of-bound-read ?s ?a)))
```

In terms of the notation above:

- O^G = {``VMware Workstation Pro/Player'', ``before 12.5.3'', ``SVGA driver''}
- O^L = {``VMware Workstation Pro/Player'', ``12.x''}
- $O^G \cap O^L$ = {``VMware Workstation Pro/Player''}

Therefore, object recall is $1/3$ and object precision is $1/2$. Likewise, consider the sets of predicates below where "_" is used to denote an argument:

- P^G = {(system _), (attacker _), (has_component _ _), (has_version _ _), (manipulates _ _), (has_no_graphics_card _), (has_no_graphics_drivers _), (crash-vm _ _), (trigger-out-of-bound-read _ _)}
- P^L = {(system _), (attacker _), (has_component _ _), (has_version _ _), (has_no_graphics_card _), (has_no_graphics_drivers _), (crash-vm _ _), (trigger-out-of-bound-read _ _)}
- $P^G \cap P^L$ = {(system _), (attacker _), (has_component _ _), (has_version _ _), (has_no_graphics_card _), (has_no_graphics_drivers _), (crash-vm _ _), (trigger-out-of-bound-read _ _)}

Therefore, predicate recall is 8/9 and predicate precision is 8/8. That is, the extracted LLM missed one predicate but the ones that it contained had the right form.

Finally, for logic edit distance, note that the extracted operator missed a disjunction in the preconditions. That's an edit distance of 1.

Our approach to evaluating and improving extraction is to ingest CVEs, use the current prompt to extract planning operators for a random subset of them, edit the operators into gold standard form, and compute the metrics. Editing extracted operators has the advantage of standardizing on predicate and variable names. As errors are discovered through this process, the prompt is refined and evaluation is repeated with the hope of leading to fewer errors in the future.

Empirical Results

This section reviews 10 CVEs and the planning operators extracted by CLLaMP to understand the kinds of errors that it commits, and to highlight the need for guidelines on what to include in operators and how. Each CVE/action pair is shown, with a discussion of the errors found in the action and ways of addressing them in the prompt. The 10 CVEs were chosen at random from the entire collection of more than 200k total CVEs.

CVE-2018-25041

"A vulnerability was found in uTorrent. It has been rated as critical. Affected by this issue is some unknown functionality of the component JSON RPC Server. The manipulation leads to privilege escalation. The attack may be launched remotely. The exploit has been disclosed to the public and may be used. It is recommended to upgrade the affected component."

```
(:action EXPLOIT-CVE-2018-25041
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'uTorrent')
                     (has_version ?s 'unknown')
                     (manipulates ?a 'component JSON RPC Server'))
  :effect (and (gain-access ?a ?s 'privilege escalation'))) \\
```

The translation above is correct except for the inclusion of the (has_version ?s unknown) predicate. It is true that there is no version mentioned for uTorrent, but the LLM is picking up on the use of the word "unknown" which was mentioned in relation to the functionality of the JSON RPC server. A better operator would not include any mention of versions that are not mentioned in the CVE, allowing systems with uTorrent to match regardless of the version. Including CVE examples in the prompt (shown in Appendix A) with no explicit version numbers should help here, although the current prompt uses about a third of GPT 3.5's available context. The operator has an object precision of 1, and object recall of 3/4, a predicate precision/recall of 1 and 6/7, and a logic edit distance of 0.

CVE-2022-47474

"In telephony service, there is a missing permission check. This could lead to local information disclosure with no additional execution privileges needed."

```
(:action EXPLOIT-CVE-2022-47474
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_local_access ?a ?s)
                     (has_component ?s 'telephony service')
                     (missing_permission_check ?s))
  :effect (and (gain-access ?a ?s 'local information disclosure')))
```

The CVE itself is so nonspecific that the operator may not be useful. The LLM invented two predicates not in any of the prompts ("has_local_access" and "missing_permission_check"). The operator has an object precision/recall of $2/3$ and 1 and a predicate precision/recall of 2/3. The logic edit distance is 0.

CVE-2005-1636

"mysql_install_db in MySQL 4.1.x before 4.1.12 and 5.x up to 5.0.4 creates the mysql_install_db.X file with a predictable filename and insecure permissions, which allows local users to execute arbitrary SQL commands by modifying the file's contents."

```
(:action EXPLOIT-CVE-2005-1636
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_local_access ?a ?s)
                     (has_component ?s 'MySQL')
                     (has_version ?s '4.1.x' 'before 4.1.12'
                       'or' '5.x' 'up to 5.0.4')
                     (modifies_file ?a 'mysql_install_db.X')
                     (insecure_permissions ?a 'mysql_install_db.X'))
  :effect (and (execute-sql ?a ?s)))
```

The translation above invents several predicates ("modifies," "insecure," and "execute") not found in the prompt. The versioning should be broken out into a logical "or" rather than trying to push it into a single has_version predicate. Including examples of CVEs with multiple, disjoint versions affected could help with the versioning disjunction. The object precision/recall is 5/6 and 5/7, the predicate precision/recall is 4/7 and 1/2, and the logic edit distance is 2 (the disjunction for the verisions, plus a conjunction for the 5.x up to 5.0.4 constraint).

CVE-2023-20922

"In setMimeGroup of PackageManagerService.java, there is a possible crash loop due to resource exhaustion. This could lead to local denial of service with no additional execution privileges needed. User interaction is not needed for exploitation. Product: AndroidVersions: Android-11 Android-12 Android-12L Android-13Android ID: A-237291548"

```
(:action EXPLOIT-CVE-2023-20922
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_component ?s 'PackageManagerService.java')
                     (has_version ?s 'Android-11')
                     (has_version ?s 'Android-12')
                     (has_version ?s 'Android-12L')
                     (has_version ?s 'Android-13')
                     (manipulates ?a 'setMimeGroup'))
  :effect (and (gain-access ?a ?s 'local denial of service')))
```

The Android versions are listed as a conjunction, and would need to be refactored into a logical "or". "PackageManageService.java" and "setMimeGroup" are descriptions of the attack mechanism, and would be present on any Android device. Omitting them from the action would be easier than populating the planning state with such fine-grained components. All objects and predicates are present, giving the operator perfect object and predicate precision and recall, but a logic edit distance of 1.

CVE-2020-0838

"An elevation of privilege vulnerability exists when NTFS improperly checks access, aka 'NTFS Elevation of Privilege Vulnerability'. "

```
(:action EXPLOIT-CVE-2020-0838
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_component ?s 'NTFS')
                     (improperly_checks_access ?s))
  :effect (and (gain-privilege ?a ?s)))
```

This is another CVE where the vagueness of the description is a challenge. However, the novel `improperly_checks_access` predicate should be omitted or replaced with `manipulates`, and the `gain-privilege` predicate should be replaced with `gain-access` as in the example CVEs. This gives an object precision or 1/3, an object recall of 1, and a predicate precision/recall of 3/5. The logic edit distance is 0.

CVE-2018-15459

"A vulnerability in the administrative web interface of Cisco Identity Services Engine (ISE) could allow an authenticated, remote attacker to gain additional privileges on an affected device. The vulnerability is due to improper controls on certain pages in the web interface. An attacker could exploit this vulnerability by authenticating to the device with an administrator account and sending a crafted HTTP request. A successful exploit could allow the attacker to create additional Admin accounts with different user roles. An attacker could then use these accounts to perform actions within their scope. The attacker would need valid Admin credentials for the device. This vulnerability cannot be exploited to add a Super Admin account."

```
(:action EXPLOIT-CVE-2018-15459
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s
                       'Cisco Identity Services Engine (ISE)')
                     (has_version ?s 'unknown')
                     (manipulates ?a 'administrative web interface')
                     (has_valid_credentials ?a ?s 'Admin'))
  :effect (and (gain-privileges ?a ?s 'additional Admin accounts')))
```

Most of the CVE is a detailed description of the attack and its consequences. In extracting an operator, the LLM invents a version string of 'unknown', and adds a new predicate, potentially useful predicate for `has_valid_credentials`. The effect predicate is again changed out, too. The operator as an object precision of 1 and recall of 3/5, and a predicate precision and recall of 3/5. The logic edit distance is 0.

CVE-2021-40309

"A SQL injection vulnerability exists in the Take Attendance functionality of OS4Ed's OpenSIS 8.0. allows an attacker to inject their own SQL query. The cp_id_miss_attn parameter from TakeAttendance.php is vulnerable to SQL injection. An attacker can make an authenticated HTTP request as a user with access to "Take Attendance" functionality to trigger this vulnerability."

```
(:action EXPLOIT-CVE-2021-40309
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s "OS4Ed's OpenSIS")
                     (has_version ?s '8.0')
                     (manipulates ?a 'cp_id_miss_attn'))
  :effect (and (gain-access ?a ?s 'SQL injection')))
```

This is one of the rare CVEs whose PDDL action fully matches the pattern used in the prompt.

CVE-2017-12167

"It was found in EAP 7 before 7.0.9 that properties based files of the management and the application realm configuration that contain user to role mapping are world readable allowing access to users and roles information to all the users logged in to the system."

```
(:action EXPLOIT-CVE-2017-12167
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_component ?s 'EAP')
                     (has_version ?s 'before 7.0.9')
                     (has_file ?s 'management.properties')
                     (has_file ?s 'application.properties')
                     (world_readable ?s 'management.properties')
                     (world_readable ?s 'application.properties'))
  :effect (and (gain-access ?a ?s 'user and role information')))
```

The `has_file` and `world_readable` predicates are novel and should be omitted, and it lacks the "manipulates" predicate of the model operators. The version specification is missing a rectriction to 7.x. The operator's object precision and recall is 3/5 and 3/7, where predicate precision and recall is 5/6 and 5/7, respectively. The logic edit distance is 1 for the version specification.

CVE-2021-46595

"This vulnerability allows remote attackers to disclose sensitive information on affected installations of Bentley MicroStation CONNECT 10.16.0.80. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file. The specific flaw exists within the parsing of 3DS files. The issue results from the lack of proper validation of user-supplied data, which can result in a read past the end of an allocated buffer. An attacker can leverage this in conjunction with other vulnerabilities to execute arbitrary code in the context of the current process. Was ZDI-CAN-15389."

```
(:action EXPLOIT-CVE-2021-46595
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'Bentley MicroStation CONNECT')
                     (has_version ?s '10.16.0.80')
                     (user_interaction_required ?s)
                     (visits_malicious_page ?a ?s)
                     (opens_malicious_file ?a ?s)
                     (parsing_3ds_files ?s)
                     (lack_of_validation ?s)
                     (read_past_end_of_buffer ?s))
  :effect (and (execute\_arbitrary\_code ?a ?s)))
```

The LLM generates a number of novel predicates for this action, mostly keying off the detailed exploit description. Even if the predicates were used, however, the roles are confused, as the victim and not the attacker must visit the malicious web page. The operator has an object precision and recall of 1/2 and 1, and a predicate precision/recall of 5/7 and 5/12. The logic edit distance is 0.

CVE-2001-0992

"shopplus.cgi in ShopPlus shopping cart allows remote attackers to execute arbitrary commands via shell metacharacters in the "file" parameter."

```
(:action EXPLOIT-CVE-2001-0992
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'ShopPlus shopping cart')
                     (manipulates ?a 'parameter file'))
  :effect (and (gain-access ?a ?s 'execute arbitrary commands')))
```

This action appropriately drops the version predicate, which isn't specified in the CVE text. Object and predicate precision and recall are all 1, and logic edit distance is 0.

Result Metrics

Overall, the average object precision and recall is .79 and .85, while the average predicate precision and recall is .8 and .74. The average logic edit distance is .4. Only two of the ten operators did not need edits.

Discussion and Future Work

This paper described and evaluated CLLaMP, a system that uses large language models to extract planning operators in PDDL format from CVEs. The goal of CLLaMP is to support reasoning about attacker plans in cybersecurity domains. The system was evaluated against gold standard planning operators using several metrics related to precision, recall, and edit distance defined for structured PDDL representations. Empirical results showed the potential effectiveness of the approach. Given the level of accuracy, the LLM could function as a code assistant to help users translate from the CVE to PDDL.

There is significant future work that can be done. This includes improving the accuracy of the translating, which a necessary component to scale up extraction of PDDL from all of the 200K+ currently available CVEs. Potential improvements could come from trying other LLMs such as Hugging Face's StarCoder which was trained to focus on coding tasks, refinements to the prompt in response to errors over larger sets of extracted operators. Other required tasks include mapping string representations of objects to tokens as required by PDDL syntax, and merging predicate and object names to standardize terms (e.g., cross-site scripting vs XSS vs basic cross-site scripting).

Appendix A: LLM Prompt

I want to convert CVEs into planning operators in PDDL. The preconditions will test whether a system is vulnerable to the attack described in the CVE. The effects will indicate what the attacker gains.

I'm going to show you example CVEs and their corresponding planning operators in the format I want.

Below is an example CVE:

CVE-2023-2387

A vulnerability classified as problematic was found in Netgear SRX5308 up to 4.3.5-3. Affected by this vulnerability is an unknown functionality of the file scgi-bin/platform.cgi?page=dmz_setup.htm of the component Web Management Interface. The manipulation of the argument winsServer1 leads to cross-site scripting. The attack can be launched remotely. The exploit has been disclosed to the public and may be used. The identifier VDB-227665 was assigned to this vulnerability. NOTE: The vendor was contacted early about this disclosure but did not respond in any way.

Here is the corresponding planning operator:

```
(:action EXPLOIT-CVE-2023-2387
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'Netgear SRX5308')
                     (has_version 'Netgear SRX5308' 'up to 4.3.5-3')
                     (manipulates ?a 'argument winsServer1'))
  :effect (and (gain-access ?a ?s 'cross site scripting')))
```

Here is another CVE:

CVE-2023-2738

A vulnerability classified as critical has been found in Tongda OA 11.10. This affects the function actionGetdata of the file GatewayController.php. The manipulation leads to unrestricted upload. It is possible to initiate the attack remotely. The exploit has been disclosed to the public and may be used. The identifier VDB-229149 was assigned to this vulnerability. NOTE: The vendor was contacted early about this disclosure but did not respond in any way.

Here is the corresponding planning operator:

```
(:action EXPLOIT-CVE-2023-2738
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'Tongda OA')
                     (has_version 'Tongda OA' '11.10')
                     (manipulates ?a 'actionGetdata'))
  :effect (and (gain-access ?a ?s 'unrestricted upload')))
```

Here is another CVE:

CVE-2023-2368

A vulnerability was found in SourceCodester Faculty Evaluation System 1.0. It has been declared as critical. This vulnerability affects unknown code of the file index.php?page=manage_questionnaire. The manipulation of the argument id leads to sql injection. The attack can be initiated remotely. The exploit has been disclosed to the public and may be used. The identifier of this vulnerability is VDB-227644.

Here is the corresponding planning operator:

```
(:action EXPLOIT-CVE-2023-2368
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s
                       'SourceCodester Faculty Evaluation System')
                     (has_version
                       'SourceCodester Faculty Evaluation System'
                       '1.0')
                     (manipulates ?a 'argument id'))
  :effect (and (gain-access ?a ?s 'sql injection')))
```

Here is another CVE:

CVE-2023-2041

A vulnerability classified as critical was found in novel-plus 3.6.2. Affected by this vulnerability is an unknown functionality of the file /category/list?limit=10&offset=0&order=desc. The manipulation of the argument sort leads to sql injection. The attack can be launched remotely. The exploit has been disclosed to the public and may be used. The associated identifier of this vulnerability is VDB-225919. NOTE: The vendor was contacted early about this disclosure but did not respond in any way.

Here is the corresponding planning operator:

```
(:action EXPLOIT-CVE-2023-2041
  :parameters (?s ?a)
  :precondition (and (system ?s)
                     (attacker ?a)
                     (has_remote_access ?a ?s)
                     (has_component ?s 'novel-plus')
                     (has_version 'novel-plus' '3.6.2')
                     (manipulates ?a 'argument sort'))
  :effect (and (gain-access ?a ?s 'sql injection')))
```

Now convert the following CVE to a planning operator.

References

2024. CVE. https://www.cve.org/. Accessed: 2024-03-05.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; and Sui, Z. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Fox, M.; and Long, D. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, 20: 61–124.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2024. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36.

Hoffmann, J. 2015. Simulated penetration testing: From"Dijkstra" to "Turing test++". In *Proceedings of the international conference on automated planning and scheduling*, volume 25, 364–372.

Jin, M.; Shahriar, S.; Tufano, M.; Shi, X.; Lu, S.; Sundaresan, N.; and Svyatkovskiy, A. 2023. Inferfix: End-to-end program repair with LLMs. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1646–1656.

Korban, C. A.; Miller, D. P.; Pennington, A.; and Thomas, C. B. 2017. APT3 adversary emulation plan. Technical report, MITRE.

Kouremetis, M.; Lawrence, D.; Alford, R.; Cheuvront, Z.; Davila, D.; Geyer, B.; Haigh, T.; Michalak, E.; Murphy, R.; and Russo, G. 2024. Mirage: cyber deception against autonomous cyber attacks in emulation and simulation. *Annals of Telecommunications*.

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. 2024. Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Oswald, J.; Srinivas, K.; Lee, J.; Katz, M.; and Sohrabi, S. 2024. Large Language Models as Planning Domain Generators. In *The 34th International Conference on Automated Planning and Scheduling*. AAAI.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Notice