CVE representation to build attack positions graphs

Manuel Poisson, Valérie Viet Triem Tong, Gilles Guette, Frédéric Guihéry, Damien Crémilleux

CVE representation to build attack positions graphs

Manuel Poisson
AMOSSYS, CentraleSupélec, CNRS, Inria, Univ. Rennes, IRISA
manuel.poisson@irisa.fr

Valérie Viet Triem Tong
CentraleSupelec, CNRS, Inria, Univ. Rennes, IRISA
valerie.viettriemtong@centralesupelec.fr

Gilles Guette
Univ. Rennes, CNRS, Inria, IRISA
gilles.guette@univ-rennes.fr

Frédéric Guihéry
AMOSSYS
frederic.guihery@amossys.fr

Damien Crémilleux
AMOSSYS
damien.cremilleux@amossys.fr

***Abstract*—In cybersecurity, CVEs (Common Vulnerabilities and Exposures) are publicly disclosed hardware or software vulnerabilities. These vulnerabilities are documented and listed in the NVD database maintained by the NIST. Knowledge of the CVEs impacting an information system provides a measure of its level of security. This article points out that these vulnerabilities should be described in greater detail to understand how they could be chained together in a complete attack scenario. This article presents the first proposal for the CAPG format, which is a method for representing a CVE vulnerability, a corresponding exploit, and associated attack positions.**

***Index Terms*—CVE, attack graph, exploit**

I. Introduction

Performing an audit of an information system (IS), such as an architecture audit, configuration audit or penetration test, are common approaches for mapping vulnerabilities. These vulnerabilities can be exploited by attackers to increase their control over the system and propagate towards their ultimate objectives such as for instance, data exfiltration or ransomware execution. Unfortunately, the estimation of the risk posed by combinations of vulnerabilities relies on auditors, who may overlook some attack paths. We argue here that the description of discovered vulnerabilities, in particular by CVEs and the various related formats, lacks precision to allow the automatic integration of their exploitation into an accurate and operational attack scenarios design. This paper proposes CAPG (*from CVE to Attack Positions Graph*), a new format designed to represent exploits of CVEs and then to highlight how multiple CVEs could be chained by attackers to spread themselves. The main purpose of CAPG is to highlight critical attack paths that need to be urgently monitored and corrected.

This article presents two main contributions. The first contribution is a new CAPG format allowing to represent exploits of CVEs. CAPG gives a precise representation of how a vulnerability can be exploited by an attacker, and more importantly, what is the gain for an attacker who has successfully exploited a given CVE. The second contribution is a methodology to populate this format by executing an exploit applicable to the CVE described. Section II presents the background and the related work and motivates the use of this new format. Section III specifies CAPG and Section IV explains how to fill it, and presents an example of its use in building an attack scenario.

II. Background and related work

A. Series of formats to deal with vulnerabilities

Common Vulnerabilities and Exposures (CVEs) are vulnerabilities that are publicly disclosed to protect against their exploitation. All CVEs are indexed in the (US) national vulnerability database (NVD) [1]. Penetration testers and attackers usually start by performing a reconnaissance of the information system they intend to infiltrate. This phase includes a vulnerability scan where tools like Nessus [2] or OpenVAS [3] can reveal the presence of well-known vulnerabilities identified by a CVE-Id. Many tools and knowledge bases use the CVE-Ids to uniquely identify a CVE. For instance, Nessus can indicate if known exploits applicable to a given CVE are available. It can be used to search for additional information on identified vulnerabilities. The next step typically consists of gathering data related to these CVEs to understand and assess the associated risks. In the NVD, a CVE is described by a free-form text description, and some additional resources depicted below. As an example, for CVE 2021-38648 the following information is included as shown in Figure 1.

Severity of a CVE with the CVSS: The Common Vulnerability Scoring System (CVSS), maintained by FIRST [4], intends to help prioritize responses and remediation. The CVSS score is computed over a CVSS vector depicting how the CVE may be exploited, its ease of exploitation, and its impact on the target.

Platforms vulnerable to a CVE with CPEs: Each CVE is linked to a list of products that are known to be affected by the CVE. These products are identified using Common Platform Enumerations (CPEs) [5], a structured naming scheme for systems, hardware, and applications. A CPE lists data like the type, the vendor, and the version of a product.

Typical weaknesses responsible for the CVE: In the NVD, CVEs are mapped to Common Weakness Enumerations (CWEs) [6] which define categories of weaknesses. CWEs are classified among over 900 types ranging from buffer overflows to cross-site scripting including hard-coded credentials, and insecure random numbers generators.

Data found in the NVD enables to estimate the security level of a system [7] but does not allow understanding of how the discovered vulnerabilities can be exploited in a complete attack

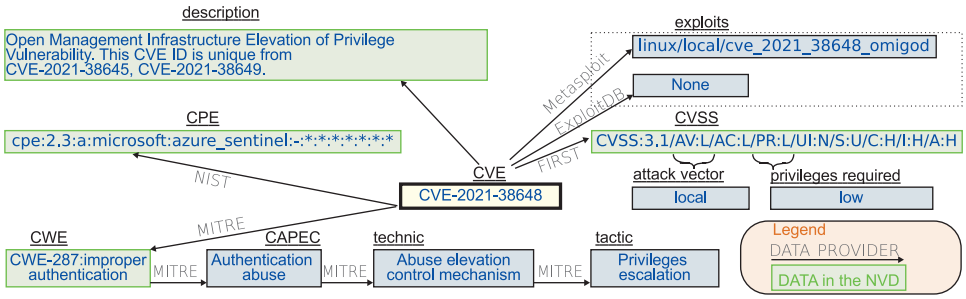


Fig. 1: `CVE-2021-38648` and its related data

scenario. We argue that highlighting a full attack scenario based on the discovery of vulnerabilities requires precise information indicating both the pre-conditions for exploiting the vulnerability and the foothold attackers will have if they succeed in exploiting it. In this context, the most useful information is given in the CVSS vector. The CVSS vector of `CVE-2021-38648` carries the information that the privileges required (*PR*) to exploit this CVE are *Low* (*L*) and that the attack vector (*AV*) is *Local* (*L*). It means that there are no special conditions for exploiting the vulnerability except the ability to run commands directly on the vulnerable system (e.g. through physical access or SSH) and control over a local account. This data is too imprecise to be integrated into an operational scenario. The open data providing the most operational information is the potential exploit linked to the CVE. An exploit is a piece of code implemented to take advantage of a vulnerability.

Metasploit [8], ExploitDB [9], and GitHub [10] are three major public exploit sources [11]. However, limitations are that not all exploits are public and public exploits are not always related to a CVE and vice-versa. A deep analysis of an exploit should provide, the conditions that attackers have to fill to use it. Unfortunately, the source code of exploits follows no particular specification and their metadata is not always consistent. For example, in Metasploit, both exploits related to the CVEs `2021-38648` and `2022-39952` lead to a session with the user `root` following their documentation, but their metadata indicating if the user reached after their full execution is "*Privileged*" or not is respectively set to "*No*" and "*Yes*". Also, the source code of exploits is not enough to automatically derive neither the context in which they can be used nor the gain from their successful execution.

The lack of an appropriate format makes it very hard to find comprehensive data to derive pre/post conditions of a CVE exploitation.

B. Related work

Some research papers and tools intend to make sense of CVEs and their related data. BRON [12] links data like CPE, CVE, CWE, CAPEC, MITRE Techniques, and Tactics by providing a bidirectional graph relating these concepts after parsing data from the NVD and MITRE ATT&CK. Similarly, other researchers [13] exploit the capabilities of the Resource Description Framework (RDF) to build a knowledge graph serving the same purpose. These graphs answer questions like: "*Which Techniques are related to CVE-2022-36804?*".

Concepts more abstract than CVE can be used to reason in attack graphs. Some researchers developed a tool to visualize how CAPECs could be chained in an information system containing CVEs [14]. They manually describe the characteristics and settings of the IS and input some data from the CVSS vector related to the CVEs. Authors of [15] even define a new ontology to describe an attack graph. Nevertheless, they give no methodology about how to fill this ontology, letting think that everything must be hand-made by human experts with good knowledge of the system and its vulnerabilities.

Other studies try to derive the causes and effects of CVEs based on their related description [16]. They propose algorithms using keywords and heuristics to extract semantics from the description of the CVE. However, the lack of standardization of CVEs' descriptions complicates the task. Others deal with the problem of CVE enrichment by automatically assigning labels to CVEs based on their descriptions and using natural language processing techniques [17].

Let us now imagine a simple realistic IS composed of 2 machines, `m0` and `m1` where a scan has revealed 3 CVEs. First, the machine `m0` is running Apache Log4j2 2.0, vulnerable to `CVE-2021-44228`, and Open Management Infrastructure (OMI) v1.6.8-0, vulnerable to `CVE-2021-38648`.The NVD database reports that the first CVE enables remote code execution (RCE), while the second allows privilege escalation. Following CVSS, these CVEs have a critical and high severity. Thirdly, the machine `m1` hosts `CVE-2022-36804`, which has a high severity and allows an authenticated RCE. To the best of our knowledge, no literature exists on the use of CVEs to design concrete attack scenarios targeting the concerned infrastructure.

III. CAPG FORMAT DEFINITION

CAPG relies on the key notion of attack position. This notion has been introduced in [18], [19] as a pair $(machine, user)$ designating an attacker who has compromised the account of a user $user$ on a machine $machine$. CAPG highlights the vulnerability, the source and destination attack position and the particular exploit used to move from the source to the destination attack

Listing 1: Generic CAPG

```
["CVE": "CVE-Id",
 "exploit" : "Exploit-URL"
 "vuln_class": one of ("application","operating-system","hardware"),
 "machines_constraints": sublist of [
     "same","different","same-windows-domain","same-ldap","adjacent-network","unconstrained"],
 "user_source": one of ("application","machine-local","system-or-root","directory","any-user"),
 "user_destination": one of ("application","machine-local","system-or-root","directory"),
 "users_constraints": sublist of ["same","different","same-application"] ]
```

position. CAPG is useful to build attack position graphs, which can represent how the attacker moves laterally and horizontally in an infrastructure. CAPG is composed of 7 elements: `CVE`, `exploit`, `vuln_class`, `machines_constraints`, `users_constraints`, `user_source`, `user_destination`. Listing 1 represent the generic format of CAPG.

`CVE` is the identifier of the form *YEAR-NUMBER* of the CVE. The same as the one used in the NVD [1].

`exploit` is the url of an exploit applicable to the CVE.

`vuln_class` specifies the component affected by the vulnerability. Its value can be either `"application"`, `"operating-system"` or `"hardware"`.

`machines_constraints` defines the network constraints between the 2 machines involved in the CVE exploitation: the machine involved in the source attack position, from which the exploit is executed and the machine involved in the destination attack position, reached after a successful exploitation. The latter is the machine where the CVE is located. `machines_constraints` is a list containing : `same` when the machines are the same, `different` when the machines are different, `unconstrained` when an arbitrary machine can exploit the CVE, `same-windows-domain` (resp. `same-ldap`) when the machines must belong to the same Windows domain (resp. same LDAP) and `adjacent-network` when the machines have to be into two adjacent networks.

`user_source` is a string from a predefined set of user account characteristics. `user_source` qualifies the user from which the exploit is executed. If the user exists only in the vulnerable application (e.g. an account on a website), the value is `application`. If it is a local user on a machine, this field's value is `machine-local`. If it is a user existing on multiple machines in the same IS, registered in the same LDAP or Active Directory (AD), the value is `directory`. Finally, it is `any-user`, if an arbitrary user can exploit the CVE.

`user_destination` is a string that represents the characteristics of the user controlled after successful exploitation. The possible value for this field is `system-or-root`, `application`, `machine-local` or `directory`. The semantic of these user's characteristics is the same as the one explained for the values of `user_source` even if the characterized user is not necessarily the same.

`users_constraints` defines constraints linking the user from which the CVE can be exploited (`user_source`) and the user accessed after the exploitation (`user_destination`). It is a list containing `same` when source and destination users are identical, `different` when they are different and `same-application` when source and destination users are accounts on the same application. For example, `["different", "same-application"]` indicates that source and destination users are two different user accounts existing one the same application.

IV. How to populate CAPG

In the following, we detail how to fill CAPG relative to a `cve` and a corresponding exploit `e`.

`CVE` is equal to the identifier of `cve`.

`Exploit` is the link towards `e`. For example, an exploit related to a specific CVE-Id can be found in Metasploit using the command `search cve:` of the msfconsole or in ExploitDB using the search filter of the web interface. Note that, there is not CAPG when the exploit is not available.

`vuln_class` can be filled using the `part` field (after the second colon in `CPE 2.3`) of the CPE related to the CVE. It is `application`, `operating-system` or `hardware` when `part` is `a`, `o` or `h` respectively.

To populate the other fields requires finding the source attack position (`machine-src`, `user-src`) allowing to execute the exploit `e`, and reach the destination attack position. We propose to do so by first deploying an environment (Docker container, physical or virtual machine) `machine-dst` vulnerable to `cve`. `machine-dst` is the machine involved in the destination attack position, reached after a successful exploitation of `cve`. Execution of the exploit `e` can be tried from different machines and users starting with the ones that are the least constraining for the attacker.

Finding `machine-src` will allow to fill the field **`machine_constraints`**. It can be done by varying the relationship between the machine from which the exploit is executed and the machine hosting the CVE. At first, the exploit is executed from an arbitrary machine that is unrelated from the vulnerable environment. If it succeeds, then `machines_constraints` is `[unconstrained]`. Else, the exploit's execution can be tried in other contexts. Deploy 2 machines `m-src` and `m-dst` in the same AD domain, make `m-dst` vulnerable to the CVE and execute the exploit from `m-src`. If it succeeds, then `machines_constraints` is `[different, same-windows-domain]`. Else, repeat with machines in the same LDAP, or in adjacent networks, or, the most constrained case, execute the exploit from the machine that is hosting the CVE. Testbeds representing these

network configurations with two machines could be prepared. One machine to execute the exploit and another with a CVE that could vary to get the CAPG of different CVEs.

The field **user_source** can be filled by varying the characteristics of the user who launches the exploit. First, run the exploit from a user that is unrelated to the vulnerable environment. If it succeeds, then `user_source` is `any-user`. Else, execute the exploit from a user that is registered in the same AD domain or same LDAP as the vulnerable machine. If it works, the value is `directory`. Else, try the exploitation using a normal local user account, on the vulnerable machine. In case of success, the value is `machine-local`. Otherwise, repeat with the user `root` or `SYSTEM`. The value is `system-or-root` if it works. Finally, create an account on the vulnerable application. If the exploit succeeds from this account, the value is `application`.

Once the exploit is fully executed, the goal is to identify the user reachable due to the CVE (i.e. the user involved in the destination attack position) in order to fill the field `user_destination`. In case of an exploit allowing code execution, run the command `whoami` (existing on UNIX and Windows OS). If the output is `root` or `SYSTEM`, then the value is `system-or-root`. Else, run commands like `Get-AdUser` or `ldapsearch` to list users in an AD or ldap, when they exist. If the output of the `whoami` command is in these lists, then `user_destination` is `directory`. Else, the user reached is a simple local user and the value is `machine-local`. If command execution is not possible, either the exploit leads to an `application` account or further manual investigation must be made.

The list `users_constraints` can be filled by comparing `user_source/destination` characteristics. To do so, the command `whoami` can be run by the user who executed the exploit and by the user reached thanks to the exploit. If the output is different, then the list of `user_constraints` contains `different`. It contains `same`, otherwise. In the case of `application` users, who cannot run commands, users of the application affected by the CVE must be listed to see if both source and destination users belong to it. If so,`same-application` must be added to the list's values.

a) Example 1: `CVE-2021-44228`: An example of the use of the CAPG format is depicted in Listing 2. At first, `CVE-2021-44228`, which affects Apache Log4j, is represented with a corresponding exploit allowing attackers to execute arbitrary code on a remote machine. The exploit is accessible on GitHub and can be located using the *search cve:* function in msfconsole. Its url allows to fill the field `exploit`. From the exploit, we can automatically extract the CVE-Id and pass this parameter to the framework DECRET [20] with the command `python decret.py -n 2021-44228 -r bullseye -s` to automatically deploy a Docker container vulnerable to this CVE. Then, the exploit can be executed with any users account of any machine by setting the container's IP address as the exploit's target (Exploit's configuration and use can also be scripted.). Therefore, there are no constraints on the source attack position: `machines-constraints` is `unconstrained` and `user_source` is `any-user`. The exploit execution spawns a shell on the vulnerable environment where the command `whoami` returns `tomcat`. The vulnerable container does not belong to any AD or LDAP, therefore `tomcat` is a local user: *user_destination* is *machine-local*.

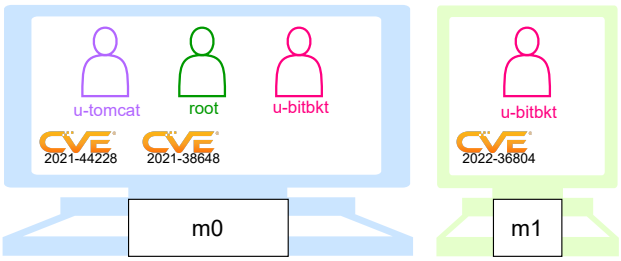
Listing 2: Three CVE represented in CAPG

```
{ "CVE": "2021-44228",
"exploit":
    "https://github.com/rapid7/metasploit-␣
    framework/blob/master/modules/exploits/␣
    multi/http/log4shell_header_injection.rb",
"vuln_class": "application",
"machines_constraints": ["unconstrained"],
"users_constraints":  ["different"],
"user_source":"any-user",
"user_destination": "machine-local"},
{ "CVE": "2021-38648",
"exploit":
    "https://github.com/rapid7/metasploit-␣
    framework/blob/master/modules/exploits/␣
    linux/local/cve_2021_38648_omigod.rb"
"vuln_class": "application",
"machines_constraints": ["same"],
"users_constraints":  ["different"],
"user_source":"machine-local",
"user_destination": "system-or-root" },
{ "CVE": "2022-36804",
"exploit":
    "https://github.com/rapid7/metasploit-␣
    framework/blob/master/modules/exploits/␣
    linux/http/bitbucket_git_cmd_injection.rb",
"vuln_class": "application",
"machines_constraints": ["unconstrained"],
"users_constraints":  ["different"],
"user_source":"application",
"user_destination": "machine-local" }
```

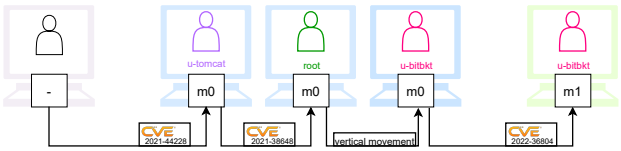
b) Example 2: `CVE-2021-38648`: The second CVE in Listing 2 represents `CVE-2021-38648` in CAPG. Values of `user_source` and `machine_constraints` unveil that this CVE can be exploited if an attacker controls a `machine-local` user on the `same` machine as the one hosting this CVE. Successful exploitation leads to a `different` user than the user in the source attack position: it leads to the user `root`. Therefore, the value of `user_destination` is `system-or-root`.

c) Example 3: `CVE-2022-36804`: . The last CVE in Listing 2 is `CVE-2022-36804`. Its `user_source` is `application` and its `user_destination` is `machine-local`. This shows that the control of a user account existing on the vulnerable application (i.e. *Bitbucket* for this CVE), allows to take control of a user existing on the machine hosting the CVE, which could lead to code execution. `machines_constraints` being `["unconstrained"]` highlights a possible exploitation from an arbitrary machine.

d) Complete attack scenario: Let's return to the small example introduced at the end of Section II. In this



(a) Attack scenario example



(b) Corresponding attack positions graph

Fig. 2: Example of an attack scenario and its corresponding attack positions graph

small information system, a machine `m0` hosts 2 CVEs `CVE-2021-44228` and `CVE-2021-38648` and second machine `m1` host `CVE-2022-36804` as depicted in Figure 2a.

We also assume *at least* the following 3 accounts: `u-tomcat` is a purely local user of `m0` able to run Apache Log4j2 2.0, `root` is a privileged user on `m0` and `u-bitbkt` is the user account of a Bitbucket repository available with the application Server and Data Center 7.0.0, executed on `m1`.

Step 1: from any attack position to (`u-tomcat`, `m0`) CAPG relative to `CVE-2021-44228` specifies that the identified `exploit` can be successfully used from any machine and with no particular initial user account (`machines_constraints` is `unconstrained` and `user_source` is `any-user`). This `exploit` gives to the attacker the control of the local user `u-tomcat` on machine `m0` since `user_destination` is `machine-local`.

Step 2: from (`u-tomcat`, `m0`) to (`root`, `m0`) CAPG relative to `CVE-2021-38648` allows to highlight that the attacker can exploit this vulnerability with the identified `exploit` from a local account, in this case `u-tomcat`, and to take control of the user `system`.

Step 3: from (`root`, `m0`) to (`u-bitbkt`, `m0`) Once the attacker has compromised the user `root` he can discover credentials of user `u-bitbkt`, who can access the Bitbucket repository hosted on `m1`.

Step 4: from (`u-bitbkt`, `m0`) to (`u-bitbkt`, `m1`) The attacker is then able to exploit `CVE-2022-36804`. CAPG relative to this last CVE shows that the attacker can run commands on `m1`.

V. Conclusion

Common Vulnerabilities and Exposures (CVEs) are exploited by attackers to progress in a compromised infratructure. Defensive tools can discover these CVEs but this article points out the lack of precision in the description of CVEs. Although very useful for measuring the overall security of an information system, knowledge of one or more CVEs does not automatically enable the construction of attack scenarios.

On the other hand, understanding a CVE exploit contains valuable information that can be used to achieve this goal. We have proposed here the new format CAPG, to describe precisely how an attacker could exploit a vulnerability. CAPG provides operational information to know if a CVE can be exploited and what is the interest of an attacker to exploit it. We also propose and test a methodology for filling in this format, based on the execution of an exploit on a suitable vulnerable environment. The main purpose of CAPG is to be used by the defense to highlight critical attack paths that need to be urgently monitored and corrected.

References

[1] "NVD - Vulnerabilities." [Online]. Available: https://nvd.nist.gov/vuln
[2] "Download Tenable Nessus Vulnerability Assessment." [Online]. Available: https://www.tenable.com/products/nessus
[3] "OpenVAS - Open Vulnerability Assessment Scanner." [Online]. Available: https://www.openvas.org/
[4] "CVSS v3.1 Specification Document." [Online]. Available: https://www.first.org/cvss/v3.1/specification-document
[5] "NVD - Products, cpe." [Online]. Available: https://nvd.nist.gov/products
[6] "CWE - Common Weakness Enumeration." [Online]. Available: https://cwe.mitre.org/index.html
[7] M. A. Williams, S. Dey, R. C. Barranco, S. M. Naim, M. S. Hossain, and M. Akbar, "Analyzing Evolving Trends of Vulnerabilities in National Vulnerability Database," in 2018 International Conference on Big Data.
[8] "Metasploit | Penetration Testing Software, Pen Testing Security." [Online]. Available: https://www.metasploit.com/
[9] "OffSec's Exploit Database Archive." [Online]. Available: https://www.exploit-db.com/
[10] "Github home page." [Online]. Available: https://github.com
[11] A. D. Householder, J. Chrabaszcz, T. Novelly, D. Warren, and J. M. Spring, "Historical Analysis of Exploit Availability Timelines," 2020.
[12] E. Hemberg, J. Kelly, M. Shlapentokh-Rothman, B. Reinstadler, K. Xu, N. Rutar, and U.-M. O'Reilly, "Linking Threat Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations for Cyber Hunting," Feb. 2021.
[13] E. Kiesling, A. Ekelhart, K. Kurniawan, and F. Ekaputra, "The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity," in The Semantic Web – ISWC, Cham, 2019.
[14] F. zdemir Snmez, C. Hankin, and P. Malacaria, "Attack Dynamics: An Automatic Attack Graph Generation Framework Based on System Topology, CAPEC, CWE, and CVE Databases," Computers & Security.
[15] K. Falodiya and M. L. Das, "Security Vulnerability Analysis using Ontology-based Attack Graphs," in 2017 INDICON.
[16] M. Urbanska, I. Ray, A. E. Howe, M. Roberts, and F. Collins, "Structuring a Vulnerability Description for Comprehensive Single System Security Analysis," 2012.
[17] D. Gonzalez, H. Hastings, and M. Mirakhorli, Automated Characterization of Software Vulnerabilities, Sep. 2019.
[18] M. Poisson, V. Viet Triem Tong, G. Guette, E. Abgrall, F. Guihéry, and D. Crémilleux, "Unveiling stealth attack paths in Windows Environments using AWARE," in CSNet 2023, Montreal, Canada.
[19] A. Berady, M. Jaume, V. Viet Triem Tong, and G. Guette, "PWNJUTSU: A dataset and a semantics-driven approach to retrace attack campaigns," IEEE Transactions on Network and Service Management, 2022.
[20] "DEbian Cve REproducer Tool (DECRET)." [Online]. Available: https://github.com/Orange-OpenSource/decret