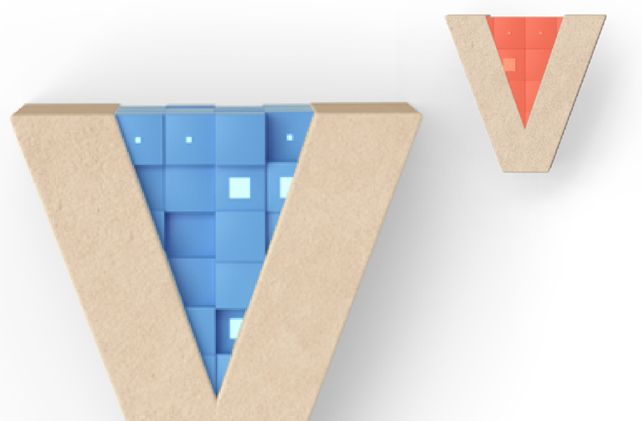
WHITEPAPER

CVE to T&TS: Using CVE attributes for MITRE ATT&CK mapping

VULCAN™

VOYAGER18

Table of contents



MITRE ATT&CK

As the cyber industry embraces and standardizes the MITRE ATT&CK framework, while at the same time understanding that vulnerability management by itself is not enough, it is required to combine both worlds and expand our visibility and perception of CVEs.

MITRE ATT&CK is a *"globally-accessible knowledge base of adversary tactics and techniques based on real-world observations"*. The ATT&CK is a dictionary that centralizes techniques that are used by threat actors along with cyber attacks. These techniques are separated and categorized into tactics, which can also be referred to as the adversary's milestones during the attack, before reaching their end goal.

An explanation of the MITRE ATT&CK framework can also be found here, in our previous blog post.

Mapping CVEs to T&Ts?

Mapping offensive techniques to CVEs allows us, the defenders, to follow the "flow" of a CVE, breaking down the steps and actions taken during its exploitation, and identifying what can be achieved (Impact) following successful exploitation.

Having this information about a CVE could help with adapting proper detections and mitigations for relevant techniques used in a wide range of CVEs within the network. Ideally, it would help detect and block techniques that are used during or following the exploitation of a CVE.

When we get deep into the ATT&CK, we can get a feeling that this mapping process is kind of artificial or synthetic. The ATT&CK framework is a collection of the adversary's actions - or steps - during the attack flow. The exploitation of a CVE might be one of these steps, but it's only one part of a larger attack journey. Naturally, ATT&CK has nothing to do with a specific CVE, which is simply a misconfiguration within a system.

For instance, buffer overflow or SQL injection are not techniques used for achieving *Execution* or *Privilege Escalation*. These are weaknesses that are caused by a misconfiguration in a vulnerable system, and exploiting these misconfigurations can be leveraged for achieving the desired tactics.

We also need to remember that the mapping is an attempt to translate a CVE in action into general coherent activities. There is no technique of ***injection*** to describe the injection of a malicious payload to a system, or ***file upload*** to describe how the attacker can upload malicious executables to an application.

What we can do is try to perceive the ***high-level behavior*** of an adversary during the exploitation of a CVE. Maybe we cannot find an injection or file upload technique, but most of the time this activity can be referred to as *Exploit Public-Facing Application (T1190).*

What has been done so far?

In the last few years, security and data researchers have tried to address the issue of mapping ATT&CK techniques to CVEs, using many different approaches.

Academic researchers, as well as security professionals from the industry, attempted to use NLP, machine learning, and deep learning methods to find a way to tackle the issue. Huge datasets that include CVE descriptions, references, and other attributes were used as a base for these methods.

Machine Learning approaches are interesting but can encounter problems. Sometimes, CVE descriptions and references are poor and do not provide enough data about the vulnerabilities, their root causes, and their consequences.

In October last year, MITRE released its methodology for mapping.

The project, which was published on GitHub last year, shows a way to describe the exploitation process and the impacts of a vulnerability. The methodology defines a logical process that each vulnerability can be passed through - mapping the exploit technique of a vulnerability and the impacts (benefits) that can be gained after successful exploitation.

The MITRE methodology includes three methods to map the techniques:
1. **Mapping vulnerability types** (SQLi, XSS, Deserialization, etc.).
2. **Mapping vulnerability functionalities** (reading files, reading from memory, overwriting existing files, etc.).
3. **Exploitation techniques** (phishing, valid hardcoded credentials, etc.).

We truly believe this methodology has a lot of potential, and we also used this as a basis for our own research at Vulcan Cyber. But the MITRE project is intended to be a manual community project in which security professionals help with mapping CVEs and contribute regularly to this knowledge base. At the time of writing, the mapping CSV file had last been updated in February 2021.

The Voyager18 approach

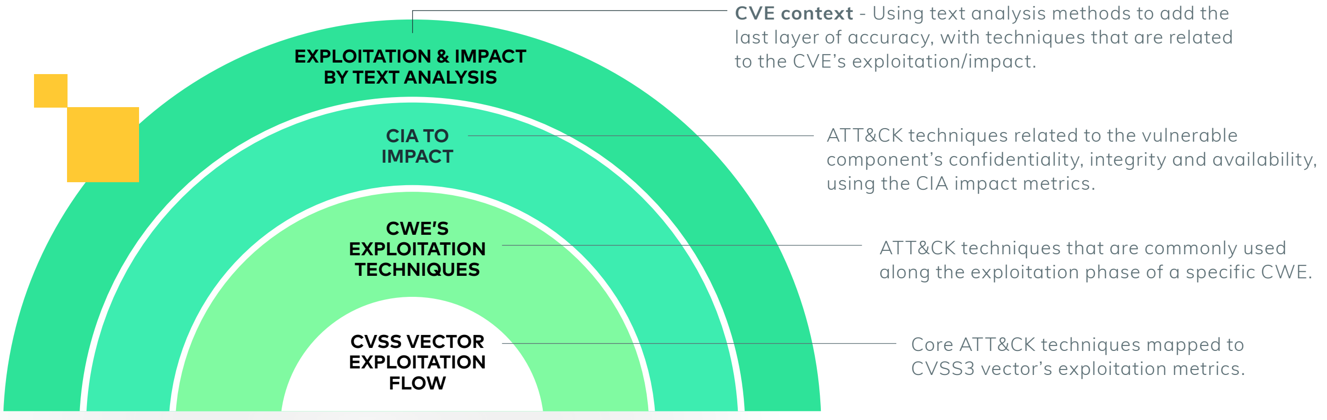
The Vulcan Cyber research team, also known as Voyager18, is a team of cyber experts working to leverage machine-learning and cyber research.

During our research, we attempted to use the data related to a specific CVE and utilize it for mapping relevant techniques to that CVE. This data includes CVE description, CWE data, and CVSS vector information.

The CWE, which is the weakness, or the root cause behind a CVE, implies the technical reason for a vulnerability's existence.

The CVSS vector is a vector representation of some valuable characteristics of a CVE, including the attack vector, the requirement for the victim interaction for successful exploitation, the requirement for the attacker to gain some sort of privileges prior to the exploitation, the CIA impact of a vulnerability, etc.

Taking this data - together with text analysis and machine learning processes that are done on the CVE's textual data (description and references) - we find **patterns** that give us indications about relevant tactics and techniques for CVEs.


EXPLOITATION & IMPACT BY TEXT ANALYSIS
CIA TO IMPACT
CWE'S EXPLOITATION TECHNIQUES
CVSS VECTOR EXPLOITATION FLOW
CVE context - Using text analysis methods to add the last layer of accuracy, with techniques that are related to the CVE's exploitation/impact.
ATT&CK techniques related to the vulnerable component's confidentiality, integrity and availability, using the CIA impact metrics.
ATT&CK techniques that are commonly used along the exploitation phase of a specific CWE.
Core ATT&CK techniques mapped to CVSS3 vector's exploitation metrics.

CWE

> *At its core, the **Common Weakness Enumeration (CWE™)** is a list of software and hardware weaknesses types. Creating the list is a community initiative aimed at creating specific and succinct definitions for each common weakness type.*

The CWE tag for a CVE could provide us with the basic data regarding a CVE. It tells us the actual weakness of the CVE - whether it's the misconfiguration or the programmer's fault - that allows the adversary to abuse a system.

By looking at the CWE of a CVE, we can get the first understanding of the vulnerability type - SQL injection, Cross-site scripting, memory vulnerability, improper permissions management weakness, and even general insufficient input validation.

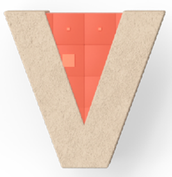
Some CWEs can immediately reveal ATT&CK techniques that are related to a CVE, with high probability.

For instance, *CWE-79* - ***Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'),*** is very straightforward. Knowing that a CVE describes a Cross-site scripting vulnerability, we can generally infer techniques that are usually related to XSS attacks, such as web session stealing, disclosure of some browser and operating-system information, defacement of the website, etc.

Mapping of CWEs like Cross-site scripting or XSS is very close in its idea to the vulnerability type mapping that was done by MITRE.

There are multiple challenges with this approach. The most obvious one we encountered so far is that not all CWEs are related to a specific known vulnerability type. Put aside XSS, SQL, and their colleagues, CWEs such as ***Information Exposure, Improper Authentication,*** or even ***Input Validation***, probably won't provide as much data about general common techniques that could be mapped to all CVEs tagged with these CVEs. The reason is that these CWEs are too "general" and gather CVEs that sometimes do not share common characteristics for ATT&CK techniques mapping.

In cases like these, we hope that with more data, such as different CVSS vectors, and textual analysis of the descriptions, we could still find patterns, or repeated behaviors, among these CWEs.



CVSS3 vector

> *The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities.*
> *The Base Score reflects the severity of a vulnerability according to its intrinsic characteristics which are constant over time and assumes the reasonable worst-case impact across different deployed environments.*

We believe that the CVSS3 vector can play a major role in the mapping. The characteristics provided by the vector bring a lot of insights when trying to understand some basic information regarding a vulnerability's exploitation phase.

Let’s begin with the basics. The following is a CVSS3 vector.

CVSS:3.1/**AV**:N/AC:L/**PR**:N/**UI**:N/S:U/**C**:H/**I**:H/**A**:H

CVSS3 vector for CVE-2018-10759

We marked the relevant values for us now: Attack Vector (AV), Privileged Required (PR), User Interaction (UI), and the CIA (Confidentiality, Integrity, and Availability).

User interaction is very simple. This metric *“captures the requirement for a human user, other than the attacker, to participate in the successful compromise of the vulnerable component. Whether the vulnerability can be exploited solely at the will of the attacker, or whether a separate user must participate in some manner.”*

UI is tagged with ‘None’ (N) when the exploitation does not require interaction from any user, or ‘Required’ (R), when *“successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited”.* The required interaction can be opening a malicious file that was downloaded by the victim, browsing a specific malicious website, or even just performing a legitimate operation in the vulnerable system that can trigger the exploitation.

Privileged required is also quite simple, and describes the level of privileges the attacker has to gain prior to the exploitation. PR is tagged with ‘None’ (N) when an authorized actor can successfully establish the attack, or ‘Low’ (L)/’High’ (H) when privileges are required, basic or administrative correspondingly.

It becomes interesting with the ***Attack vector***, because in CVSS3.1, FIRST tries to tackle misunderstanding issues with the Attack Vector of CVSS3.0.

According to the specification, this metric *“reflects the context by which vulnerability exploitation is possible”.*

There are four options for the attack vector value, which are Network (N), Adjacent (A), Local (L), and Physical (P). In this discussion, we are focusing on Network and Local vectors because their CVEs are the majority of the four.

Network vector takes place when *“the vulnerable component is bound to the network stack”* so *“such a vulnerability is often termed “remotely exploitable”.* On first reading, this description might mislead us to think that the Network vector is tagged only for remotely exploitable services that are exposed to the network, but there’s more to it than that, as we’ll soon see.

Local vector takes place when *“The vulnerable component is not bound to the network stack”*. It happens when *“the attacker exploits the vulnerability by accessing the target* *system locally (e.g., keyboard, console), or remotely (e.g., SSH)"* or when *"the attacker relies on User Interaction by another person to perform actions required to exploit the vulnerability".* This one is straightforward - if an attacker already has access to a system, via remote management protocol or local access, it means that the attack vector is local because the vulnerable component could not be exploited directly via the network stack. In addition, in cases when the attacker can exploit the vulnerability without even accessing the system, for example by providing a malicious XLSX file to a victim, it will be considered local as well.

Due to misinterpretation by people, in CVSS3.1, the guideline was clarified. And in the user guide, the following section (3.3) was added:

> *"Guidance concerning Local attacks was improved in CVSS v3.0 by clarifying the definitions of the Network and Adjacent values of the Attack Vector metric. Specifically, analysts should only score for Network or Adjacent when a vulnerability is bound to the network stack. Vulnerabilities which require user interaction to download or receive malicious content (which could also be delivered locally, e.g., via USB drives) should be scored as Local.*
> *For example, a document parsing vulnerability, which does not rely on the network in order to be exploited, should typically be scored with the Local value, regardless of the method used to distribute such a malicious document (e.g., it could be a link to a web site, or via a USB flash drive)."*

In section 3.10 more considerations regarding Attack Vector were added:

> *"Vulnerabilities where malicious data is received over a network by one component, then* ***passed to a separate component*** *with a vulnerability should be scored with an Attack Vector of* ***local****. An example is a web browser that downloads a malicious office document, saves it to disk, and then starts a vulnerable office application which reads the saved file.*
> *In cases where the vulnerable functionality* ***is part of the component that receives the malicious data, Attack vector should be scored as Network****. An example is a web browser with a vulnerability in the browser itself, or a browser plugin or extension, that triggers when the malicious data is received."*

So, when talking about local exploitation on a client workstation, the analyst should distinguish between Network and Local by the way the malicious payload is delivered or sent to the vulnerable component.

Another confusing issue was addressed in section 3.7, where FIRST tells us how to deal with vulnerabilities in software libraries, where there are various ways the library could be used by other programs that could affect the CVSS vector differently:

> *“When scoring the impact of a vulnerability in a library, independent of any adopting program or implementation, the analyst will often be unable to take into account the ways in which the library might be used. While specific products using the library should generate CVSS scores specific to how they use the library, scoring the library itself requires assumptions to be made. The analyst should score for the* ***reasonable worst-case implementation scenario****. When possible, the CVSS information should detail these assumptions.*
> *For example, a library that performs image conversion would reasonably be used by programs that accept images from untrusted sources over a network. In the reasonable worst case, it would pass them to the library without checking the validity of the images. As such, an analyst scoring a vulnerability in the library that relates to the incoming data should assume an Attack Vector (AV) of Network (N),”*

There is a reason that a ***“reasonable worst-case implementation scenario”*** was marked in the quote. The spec instructs the analyst to score the CVSS for the reasonable worst-case scenario. which makes it a little bit harder to create an aligned standard of scoring and leaves room for the analyst’s discretion and imagination.

After reading and understanding the definitions and use-cases, we can try to find combinations and patterns that might help us to better describe the exploitation flow of a vulnerability.

As an example, with CVEs with a combination of AV:L, PR:L, and UI:N, we could clearly identify a massive amount of local privilege escalation (LPE) vulnerabilities. It makes sense - the attacker has low privileges on the system (PR:L), the vulnerability can be exploited by local access to the system (AV:L), and there is no need for any interaction by another user for successful exploitation. This pattern can assist us in mapping ***Command and Scripting Interpreter (T1059),*** which is the initial position of the attacker (e.g. Windows CMD/Powershell or Unix shell), and ***Exploitation for Privilege Escalation (T1068).***

It is not 100% accurate, and during the research, we encountered different cases that put our assumptions in doubt, but we hope these obstacles can be beaten with more work and analysis.

Confidentiality, integrity, and availability?

In the section above, we explored the exploitability metrics. Aside from these, the impact metrics can also provide valuable data that should be taken into consideration.

Let's sum up the three metrics with the definitions from the specification:

- **Confidentiality** - This metric measures the impact on the confidentiality of the information resources managed by a software component due to a successfully exploited vulnerability.
- **Integrity** - This metric measures the impact on the integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of information.
- **Availability** - This metric measures the impact on the availability of the impacted component resulting from a successfully exploited vulnerability. While the Confidentiality and Integrity impact metrics apply to the loss of confidentiality or integrity of data (e.g., information, files) used by the impacted component, this metric refers to the loss of availability of the impacted component itself.

The possible values are 'HIGH' (H) to reflect the result of a total loss of the metric, 'Low' (L) to show that there is partial or limited loss of the metric, and 'None' (N) when a metric is not affected by exploitation.

As with the exploitability metrics, the team tried to find interesting patterns when investigating CIA values of vulnerabilities and to map ATT&CK techniques to the impact described in the CVE description.

An obvious example is the CIA vector of ***'C:N/I:N/A:H'***. This vector states that there is no loss of confidentiality or integrity after exploitation of a CVE, but only for the availability. It turns out that most of the vulnerabilities', tagged with this CIA vector, reviewed during the research, showed that there is no need to textually analyze the data and look for possible impacts - the only one is ***Endpoint Denial of Service: Application or System Exploitation (T1499.004).***

We also examined the ability to infer impact techniques for CIA vectors per CWEs. Taking ***CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer***, which is a parent CWE for many memory type weaknesses (Out-of-bound read, Out-of-bound write, User after free, Heap or stack buffer overflow, etc.). While exploring CVEs tagged with CWE-119, we could notice a massive amount of privilege escalation and code execution vulnerabilities when filtering CVEs with the CIA vector of ***'C:H/I:H/A:H'.***
The CWE-119 definition page strengthens this claim, and under the ***'Common Consequences'*** paragraph they mention the technical impact or consequences of *"Execute Unauthorized Code or Commands..".*

For CWE-22 ***Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')*** with CIA vector of ***'C:H/I:N/A:N'***, a trend of unauthorized read access to filesystem impact ***(Data from Local System - T1005)*** can definitely be seen.

Text analysis

At first, we wanted to avoid going in this direction. We worked hard to find ways of mapping ATT&CK techniques to CVEs solely using patterns of CWEs and CVSS3 vectors - without leveraging any textual analysis method on the CVE description.

But we found this difficult to accomplish. Our approach is based on finding **patterns,** and these patterns are not absolute, for a number of reasons.

The main reason is that there are distinctions between CVE characteristics that cannot be differentiated and divided based on the CWE and CVSS3 vector parameters.

Let's look at CVE-2017-9964:

> *"A Path Traversal issue was discovered in Schneider Electric Pelco VideoXpert Enterprise all versions prior to 2.1.* ***By sniffing communications,*** *an unauthorized person can execute a directory traversal attack resulting in authentication bypass or session hijack."*

This is a path traversal (CWE-22) vulnerability, with a CVSS3 vector of ***'AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:H/A:N'***. In order to exploit the vulnerability, the attacker should **sniff communication,** which is naturally mapped to techniques ***Network Sniffing (T1040)*** **or** ***Adversary-in-the-Middle (T1557)***. We could not figure out that the exploitation phase of this CVE includes network sniffing without reading the description.

Another one is CVE-2021-42771:

> *"Babel.Locale in Babel before 2.9.1 allows attackers to load arbitrary locale .dat files (containing serialized Python objects) via directory traversal,* ***leading to code execution****."*

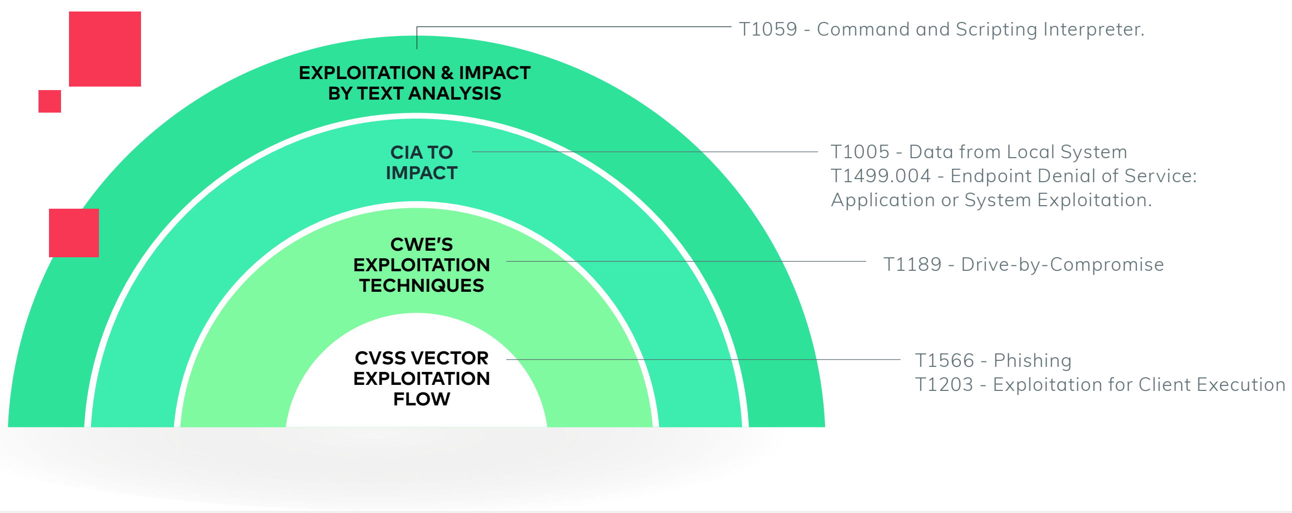
This instance is a path traversal vulnerability that leads to code execution, which is not a common impact of path traversal attacks.

In order to overcome these difficulties, we added to the mapping process some text analysis methods.
We started by extracting phrases from CVE descriptions using the RAKE (Rapid Automatic Keyword Extraction) algorithm.
Then we manually selected meaningful phrases for each CWE and grouped them into groups with similar or close semantic meanings that characterize the CWE.
These phrases could be references for more possible exploitation and impact techniques, such as ***"via crafted GIF file", "send a specially crafted packet", "via a crafted serialized Java object", "read files from the local filesystem", "leading to information disclosure"*** and many more.

Putting it all together

In the first sections, we drilled down into each phase of the mapping.
In the following diagram, we will explore an example of the whole mapping process while trying to identify possible patterns.

CVE-2016-0062 - *"Micrososft Internet Explorer 11 and Microsoft Edge allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted website, aka "Microsoft Browser Memory Corruption Vulnerability."*
CWE-119 - Improper Restriction of Operations within the Bounds of a memory Buffer.
CVSS3 vector - AV:N/AC:L/PR:N/UI:R/s:U/C:H/I:H/A:H


EXPLOITATION & IMPACT BY TEXT ANALYSIS
T1059 - Command and Scripting Interpreter.
CIA TO IMPACT
T1005 - Data from Local System
T1499.004 - Endpoint Denial of Service: Application or System Exploitation.
CWE'S EXPLOITATION TECHNIQUES
T1189 - Drive-by-Compromise
CVSS VECTOR EXPLOITATION FLOW
T1566 - Phishing
T1203 - Exploitation for Client Execution

Exploitation metrics of ***AV:N, PR:N***, and ***UI:R*** indicate vulnerability that can be exploited remotely with the requirement for the victim interaction, without the need for valid credentials or privileges prior to the exploitation.

The pattern of a memory buffer weakness with the exploitation described above, helps us determine that this is a vulnerability in a client application.

As we already mentioned, section 3.10 in the CVSS3 user guide can support this hypothesis:

> *"... In cases where the vulnerable functionality* ***is part of the component that receives the malicious data****, Attack Vector should be scored as* ***Network****. An example is a web browser with a vulnerability in the browser itself, or a browser plugin or extension, that triggers when the malicious data is received."*

It brings us to map ***T1566 (Phishing)*** and ***T1203 (Exploitation for Client Execution)***.

Using the vector and the CWE information, we can also map ***T1189 (Drive-by Compromise)*** for web browser components.

The next step is to review the CIA metrics - ***C:H/I:H/A:H***. This time, we can presume that the adversary can either obtain local data from the system or cause a crash to the system after successful exploitation. We can now map ***T1499.004 (Endpoint Denial of Service: Application or System Exploitation)*** and ***T1005 (Data from Local System).***

Lastly, we can sharpen it a bit more by analyzing the CVE description, and figuring out that another possible impact of the vulnerability is executing malicious code on the vulnerable system. It leads us to map ***T1059 (Command and Scripting Interpreter).***

From the defender's perspective, the final phase is taking the mapped techniques and using them in order to place proper detection and mitigation mechanisms.

For example, if we look at ***T1203 (Exploitation for Client Execution)***, the mechanisms are helpful in defeating the exploitation of the vulnerability. The defender can monitor the Internet Explorer application logs to pinpoint crash events after browsing a particular website ***(DS0015)***, and monitor the process by creating events to spot successful exploitation attempts ***(DS0009)***. The defender can place some mitigation techniques such as application sandboxing ***(M1048)*** or any exploit protection application ***(M1050)*** such as WDEG or EMET, to hopefully prevent the exploitation.

Future research and challenges

We're only at the beginning of our research and there are many directions and ideas left to explore.

The first is to research the capability to distinguish between different types of CPEs and to use this differentiation in order to better map ATT&CK techniques. Currently, we categorize the vulnerable component into various different types - client applications, public-facing services, shared libraries, operating systems, etc.

Another interesting research path is to check "inheritance" between CWEs, or the relationships between parents of CWEs to their children. For instance, ***CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer*** **is the parent of** ***CWE-125: Out-of-bounds Read***. It is interesting to understand what this parental relationship means in the context of inheriting ATT&CK techniques from parent to child.
There are plenty more ideas and potential research directions. Besides these, there are also challenges we need to tackle. The major one is dealing with CVEs with poor description.

Poor descriptions, without much information regarding vulnerabilities' exploitation and impact, minimize the algorithm's capability to map the right and exact techniques to a CVE. As long as CWE and CVSS3 vector patterns are not 100% accurate, textual processing and analysis will always be a part of the process. And for textual processing to succeed, the description should be informative enough.

Regarding the textual analysis, in future work, we plan to extract keywords also from ATT&CK techniques. We will tag them to the semantic groups we created with the CWE keywords for matching CWEs and ATT&CK techniques.

Another direction we can research is to isolate phrases that are unique per CWE and map them to ATT&CK techniques. Lastly, we will find overlapping phrases between CWEs, in order to apply known mapped techniques of one CVE to another one.

About Voyager18

The Vulcan Cyber research team, also known as Voyager18, is a team of cyber experts working to leverage machine-learning and cyber research to ensure Vulcan Cyber remains a cyber security leader in the field. The team's main objective is to research the latest cyber risk trends, including new attack types and remediations. The team is also responsible for bringing innovation to the Vulcan Cyber platform so that our customers get improved and customized cyber risk management capabilities. This includes research of more specific and accurate risk calculations that can truly help our customers own their risk.

SEE THE FULL RESEARCH

EXPLORE MORE

References

[1] MITRE ATT&CK Matrix - Enterprise

[2] CWE View: Research Concepts

[3] CVSS Version 3.1 - Specification Document

[4] CVSS Version 3.1 - User Guide

[5] Mapping MITRE ATT&CK to CVEs for Impact by CTID

[6] Linking CVE's to MITRE ATT&CK Techniques by Aditya Kuppa, Lamine Aouad and Nhien-An Le-Khac