EMPOWERING DEFENDERS:

HOW AI IS SHAPING MALWARE ANALYSIS

VIRUSTOTAL

NOVEMBER 23

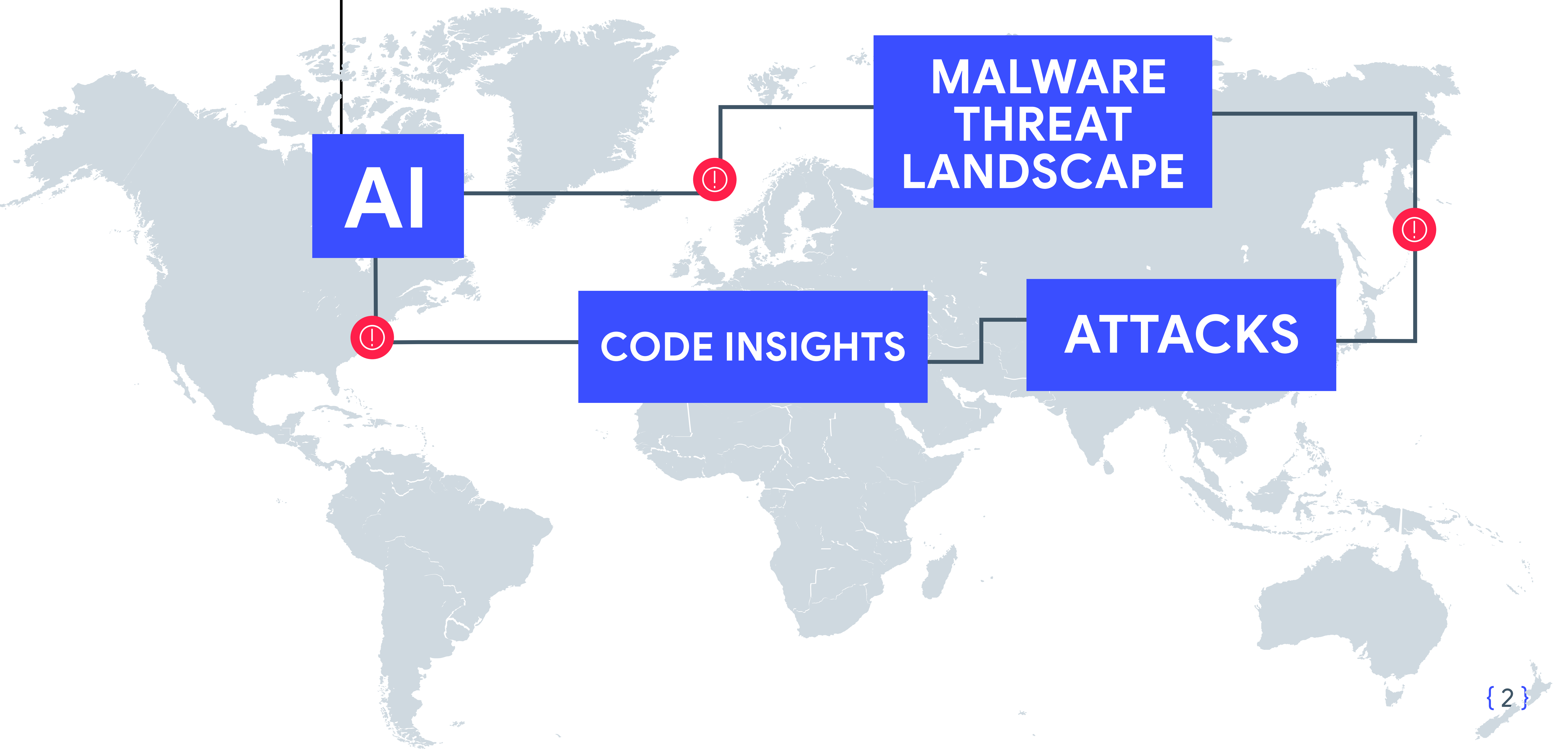
Welcome

By sharing our visibility into the **malware threat landscape**, we hope to help researchers, security practitioners, and the greater public better understand the evolution of malware attacks, AI engines' role in analyzing them, and how these solutions can help defenders.

VirusTotal announced **Code Insight** in April 2023. This was the first time we launched a feature leveraging AI for code analysis, in particular dedicated to analyzing suspicious scripts. Since then, two more AI engines, also dedicated to analyzing different scripting languages, joined VirusTotal's crowdsourced AI initiative. Since then we have tested these AI engines against hundreds of thousands of files. This report provides an overview of the impact that these AI engines have for malware analysis.

It is important to keep in mind that we are just at the very beginning of exploring all the possibilities AI offers to security. This document provides a first real test, including unexpected results, strengths and weaknesses for AI malware analysis capabilities. What is not covered in this document are other AI security-related topics such as attacks against AI systems.

We hope this report contributes to ongoing community efforts to discover and share actionable information on malware trends.



Executive Summary

Key observations:

- AI complements traditional malware analysis tools by providing a new functionality, leading to very significant time savings for cyber defenders.
- AI excels in identifying malicious scripts, particularly obfuscated ones, achieving up to 70% better detection rates compared to traditional methods alone.
- AI demonstrates enhanced detection and identification of scripts exploiting vulnerabilities, with an improvement on exploit identification of up to 300% over traditional tools alone.
- AI's ability to identify and explain malicious code in simple language could reduce the need for highly-specialized malware analysis skills in cybersecurity: helping bridge the gap of Europe's missing 500,000 cybersecurity experts.
- We observed suspicious samples using AI APIs or leveraging enthusiasm for AI products for distribution. However, AI usage in APT-like attacks cannot be confirmed at this time.

Methodology

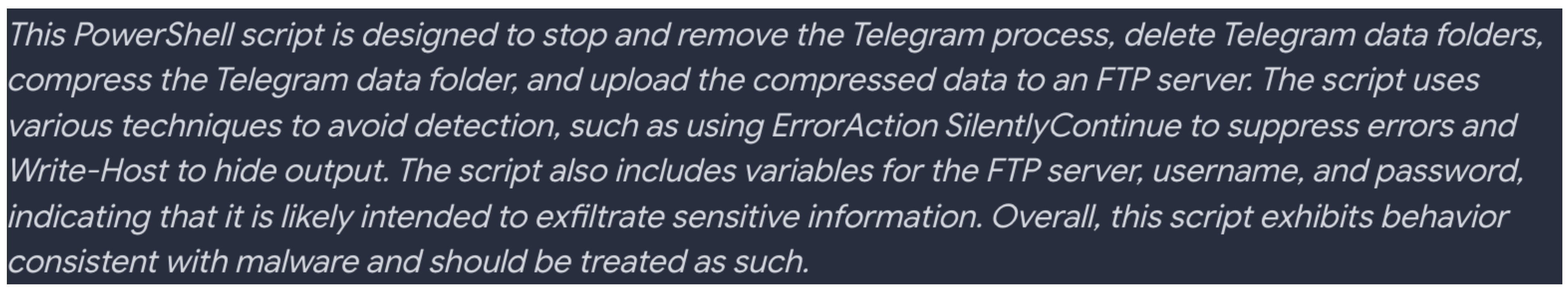
VirusTotal is in a unique position to provide comprehensive visibility across the malware landscape. Over the last 19 years, we have processed more than two million files per day **across 252 territories in 195 countries**. VirusTotal also harnesses contributions of its community of users to provide relevant attack context. We use this **crowdsourced intelligence** to analyze relevant data, share an understanding of how attacks develop, and help inform how they might evolve in the future.

VirusTotal relies on **crowdsourced contributions**, which provides a bigger picture on how different malware spreads and attacks evolve. All data in this report is compiled using a representative subset of submissions from our users from April 2023, when VirusTotal released Code Insight, until the end of October 2023. In addition to Code Insight, VirusTotal added two additional AI-based engines, all of them specialized in analyzing different scripting languages.

How AI is shaping malware analysis

The three **AI engines** implemented in VirusTotal were **designed for code analysis**, and we included them in the analysis pipeline for any suspicious script. The fantastic capability of AI engines for writing code is also reflected in their capability to "understand" it and explain in natural language.

The result of this is an incredible amount of time saved for analysts, who now can more quickly understand what the suspicious code does. There is another important angle to this: **AI engines, unlike other more traditional security tools, provide a detailed explanation instead of a "binary" verdict**, which allows human analysts to make a decision in certain gray cases, as seen in Figure 1.



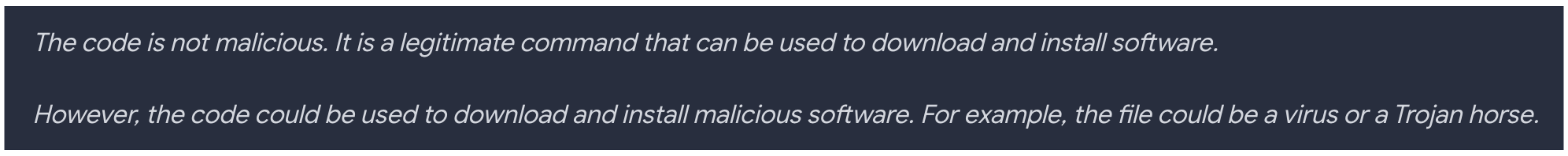
This PowerShell script is designed to stop and remove the Telegram process, delete Telegram data folders, compress the Telegram data folder, and upload the compressed data to an FTP server. The script uses various techniques to avoid detection, such as using ErrorAction SilentlyContinue to suppress errors and Write-Host to hide output. The script also includes variables for the FTP server, username, and password, indicating that it is likely intended to exfiltrate sensitive information. Overall, this script exhibits behavior consistent with malware and should be treated as such.

Fig 1.

Code Insight example

This also has significance for the future of malware analysis. Google Cloud's Cybersecurity Forecast 2024 predicts that "cyber defenders will use generative AI related technologies to strengthen detection... as well as speed up analysis and other time-consuming tasks, such as reverse engineering". Our findings are consistent with that. Malware analysis is a heavily time-consuming task and requires highly specialized knowledge and experience. **AI's ability to "understand" suspicious script and explain it in natural language** reduces not just the time taken in analyzing code, but also the level of knowledge needed to do so - making it possible, for the first time, for non-cybersecurity experts to spot and prevent malware attacks.

AI engines are also able to provide verdicts based on their descriptions of the sample's behavior. However, we found cases where AI engines' verdicts were inconsistent, and while **the technical description of the sample was accurate and good enough for understanding its behavior**, it doesn't necessarily translate well into a verdict. This regularly occurs in human analysis; when lacking enough context, sometimes it is not possible to determine the maliciousness of a file.



The code is not malicious. It is a legitimate command that can be used to download and install software.

However, the code could be used to download and install malicious software. For example, the file could be a virus or a Trojan horse.

Fig 2.

Confusing verdict

VirusTotal **aggregates results from its security ecosystem**, which includes security tools, crowdsourced intelligence, and antivirus providers. One interesting aspect of including AI engines is understanding how they complement other more traditional solutions. Although comparing AI engines with antivirus solutions when it comes to detection might be unfair, as these technologies provide very different outcomes, why and how differences occur **helps us to better understand the analysis results**.

We used small sample sets to compare AI engines and antivirus detection for "malicious" samples.

98.5%

MICROSOFT OFFICE FILES

AI engines & antivirus **agreed** on 98.5% of verdicts, with occasional fine-tuning disagreements

96.23%

POWERSHELL FILES

AI engines & antivirus showed a verdict **agreement** of 96.23%. We believe disagreements were mostly for lack of context

72.45%

PHP FILES

AI engines & antivirus showed 72.45% **agreement.** AI engines appear to be better equipped for source code analysis, including **deobfuscation** capabilities that helped to rightly identify additional malicious samples

When antivirus solutions detected something as malicious, but the AI engine's description didn't explicitly agree, we find the disagreements are more based on a difference in criteria than in AI engines making a bad call. Independently of the final AI engine's verdict, the description of the behavior of the script was correct.

AI engines showed much better performance for some types of **deobfuscation** where traditional solutions seem to struggle more. We didn't perform a full analysis for all possible obfuscation methods, but AI engines seem to be more "flexible" when finding solutions for interpreting all types of scripting and source code. Traditional solutions were only able to flag that a given sample used any type of obfuscation in 5% of the cases where AI engines detected it.

An unexpected benefit of using AI engines for script analysis was **identifying the sample's file type**. This is not a trivial problem, as text files are hard to classify. Since we implemented crowdsourced AI engines, we now recognize more than 100 text formats, including Mathematica, QML, R, Rust, Lua, and even LAMMPS (a molecular dynamics simulator).

Common Vulnerabilities and Exploits (CVE) discovery

AI engines showed a surprising capability to discover suspicious scripts exploiting vulnerabilities or identifying targets vulnerable to a specific CVE. This is because other security tools used in VirusTotal for CVE detection, including antivirus solutions and crowdsourced rules, were able **to identify CVE exploitation in only 25.6%** of the suspicious scripts correctly identified by AI engines. In the cases where AI engines and other security technologies agreed that a script implemented an exploit, they didn't agree on the particular exploited vulnerability (CVE) in 53.4% of the cases - with AI engines being right in case of disagreement.

One of the main reasons for AI engines' CVE detection capability is their ability to understand comments and references to the exploited vulnerability in the code of the analyzed script. However, we also found samples where this was not the case, and AI engines detected the vulnerability (apparently) based on the exploitation technique.

It is important to note that we were also able to identify scripts where antivirus detected a CVE, but the AI engine didn't. However, for those cases AI engines correctly described the behavior of the script and how the exploitation was implemented.

It is fair to say that the subset of AI-identified CVE-related samples does not entirely consist of malicious scripts. Many of them are vulnerability scanners, proof of concepts, defanged scripts, Metasploit modules, or even vulnerable source code. The difference between a proof of concept and a malicious script can be very blurry. For our purposes, we consider a malicious script to be any weaponized script ready to be executed with minimal or no modification (for example, a script deploying a payload).

Our criteria does not consider Metasploit modules malicious by themselves, as they are part of a framework. Surprisingly, AI engines showed very good results at identifying Metasploit modules and vulnerabilities they implement during an active attack. While antivirus detection appears to be lower at this stage, its ability to detect modules prior to an active attack highlights how these two technologies can complement each other in mitigating threats throughout an attack lifecycle.

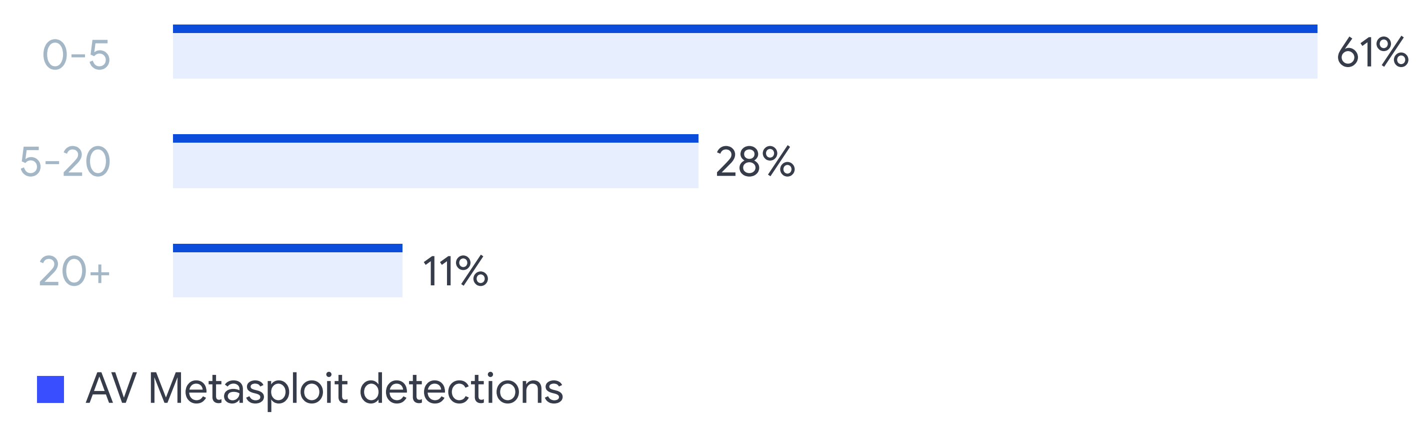


Fig 3

Metasploit AV detection

We double checked all the scripts detected by AI engines as implementing some CVE exploitation for malicious purposes (according to our previously defined criteria). For 41% of these malicious scripts, there is no antivirus detection. Figure 4 shows the antivirus detection distribution for this data set.

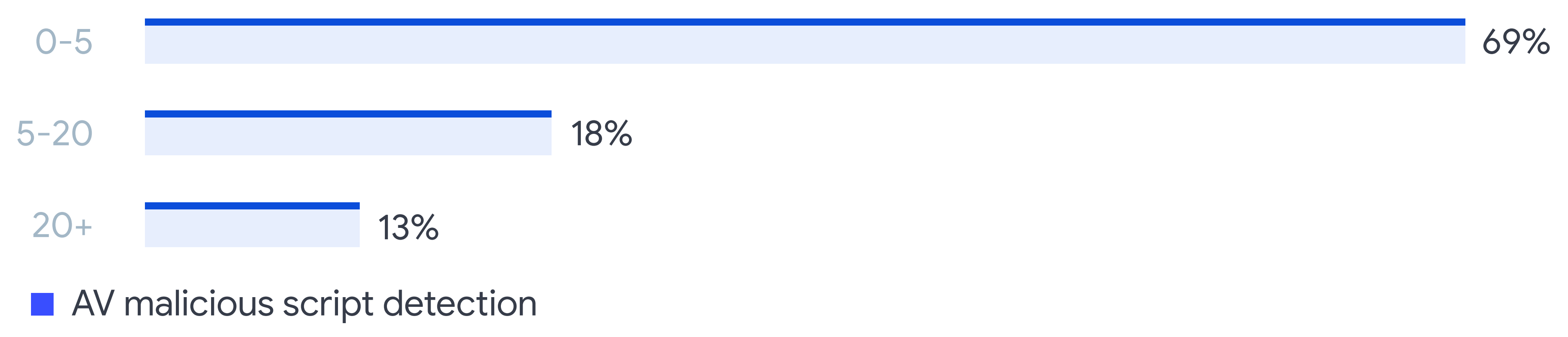


Fig 4

AV detection for scripts detected as malicious by AI engines

There is one potential explanation for this behavior. These malicious scripts are of a different nature, including automatic scanning for and exploitation of content management systems, escalation of privileges in non-Windows systems, abuse of IoT devices, and more. One could argue that this type of malicious behavior is not a top priority for some antivirus engines, which are traditionally more focused on endpoint protection. Although our analysis focused on CVE detection by AI engines, the formidable capabilities shown for script analysis represent a powerful tool for **helping detect and secure against a traditionally overlooked set of malicious tools**.

Not only that, we believe there is a clear room for improvement as the models get more contextual data. For instance, it seems all gray cases where AI engines "hesitate" whether a script downloading and executing software should be considered malicious or as an installer, could be solved by providing more details on the different elements involved, such as the URL and downloaded binary.

AI-generated malware

The question most asked of VirusTotal and our team since AI became more mainstream is "have you found any AI generated malware". Detecting if any malware was "AI generated" is a challenging task. How does one trace where any source code comes from? We played with different ideas, trying to find unusual patterns in malware families and actors for the last 12 to 15 months. In particular, we spent some time on North Korean attributed actors, given recent claims on the use of AI engines for writing malicious software by a U.S. government advisor. Through all of our research, we didn't see any strong indicators.

We found some underground offerings of malware generators claiming to use AI engines for sample creation, but none of the ones we analyzed worked properly. There is also a marketplace of "uncensored" AI engines offered in such underground markets for different criminal activities, including malware creation and assisting Business Email Compromise (BEC) attacks. This is something we will study more in future research.

We were expecting attackers to leverage AI engines for all social engineering activities. Although some isolated examples were detected, our expectation was on a more global scale, especially for extending phishing attacks to a broader targeting group (for instance, expanding the number of languages used). For the moment, we haven't observed any anomalies in terms of phishing distribution.

What we found are different malware families using AI themes for distribution. This is not surprising, given attackers' opportunistic nature on trending topics.

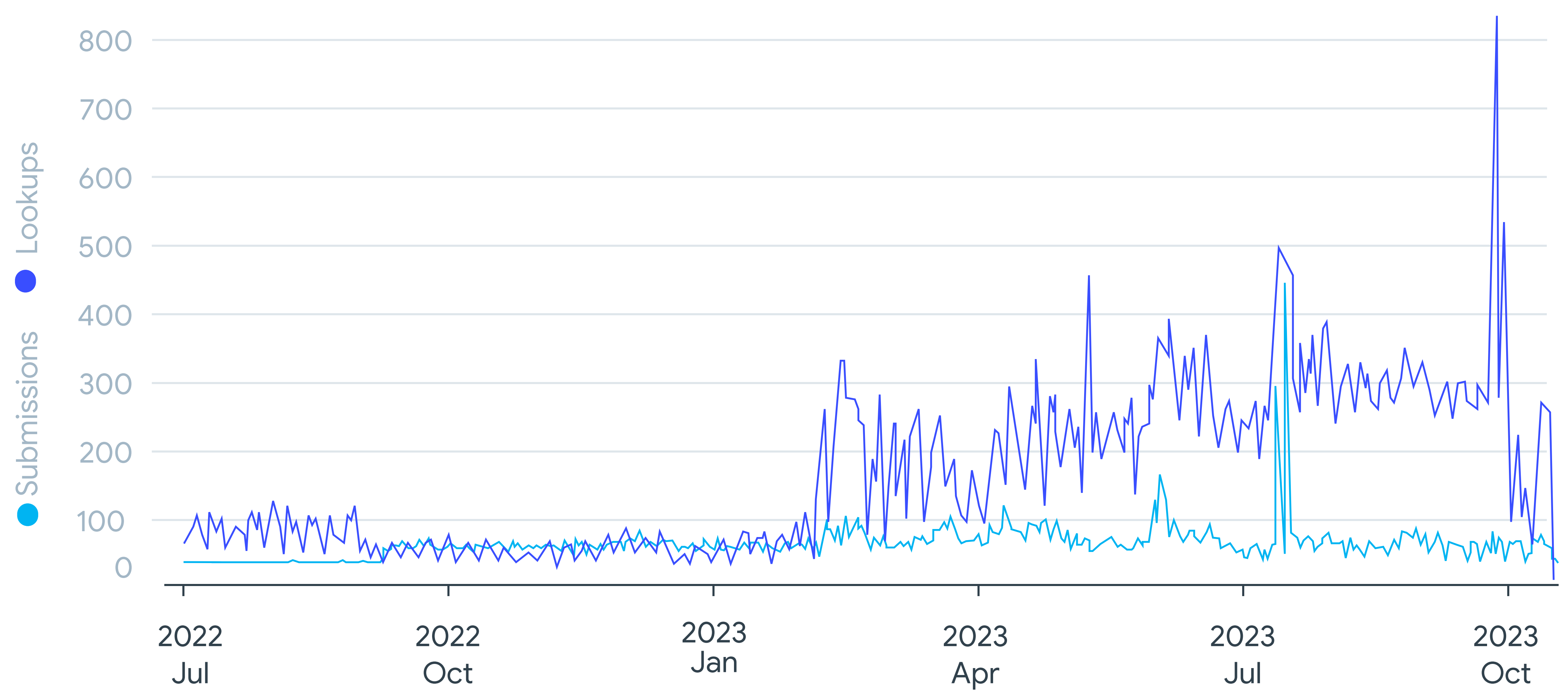


Fig 5

Evolution of AI-disguised malware

Figure 5 shows malicious samples disguised (using the same icons, names or metadata) as either Bard or ChatGPT for its distribution. We can observe a growing number of samples submitted to VirusTotal since the beginning of 2023, with a peak around July-August, caused by a single adware campaign that used hundreds of product names (including "ChatGPT Desktop Application") as bait for distribution. The number of lookups in the platform for these samples, which is a good indicator of how spread out they are, shows a slowly growing trend during the year.

Most of these disguised samples (88%) are trojans targeting Windows systems, followed by Android samples. Families distributed using this method include DCRat, RedLine, and Chaos Ransomware.

We also found a multitude of domains and URLs using ChatGPT icon as favicon for malicious purposes such as malware distribution. In some cases, we found targeted malware interacting with models hosted in legitimate AI-related domains (huggingface).

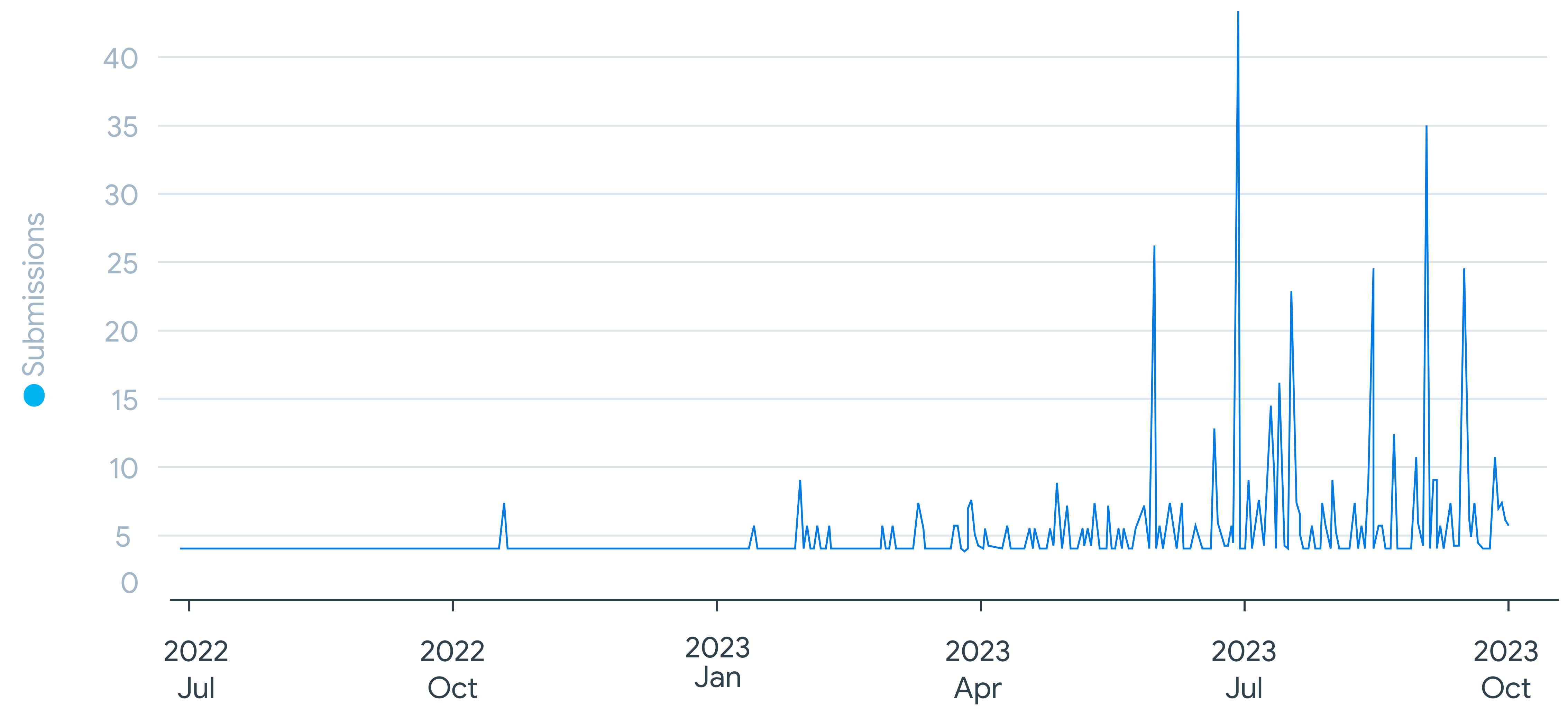


Fig 6

Evolution of AI-disguised URLs

Even more interesting than malware disguised as AI products, we wanted to explore any malware interacting with OpenAI's API, which can be used for different purposes. In terms of malware dynamic behavior, BlackMamba illustrates how it could be used on a very technical level; however, we didn't find any sample with such behavior.

In 2023, we observed a growing number of submissions of malicious files having traces of interacting with the OpenAI API.

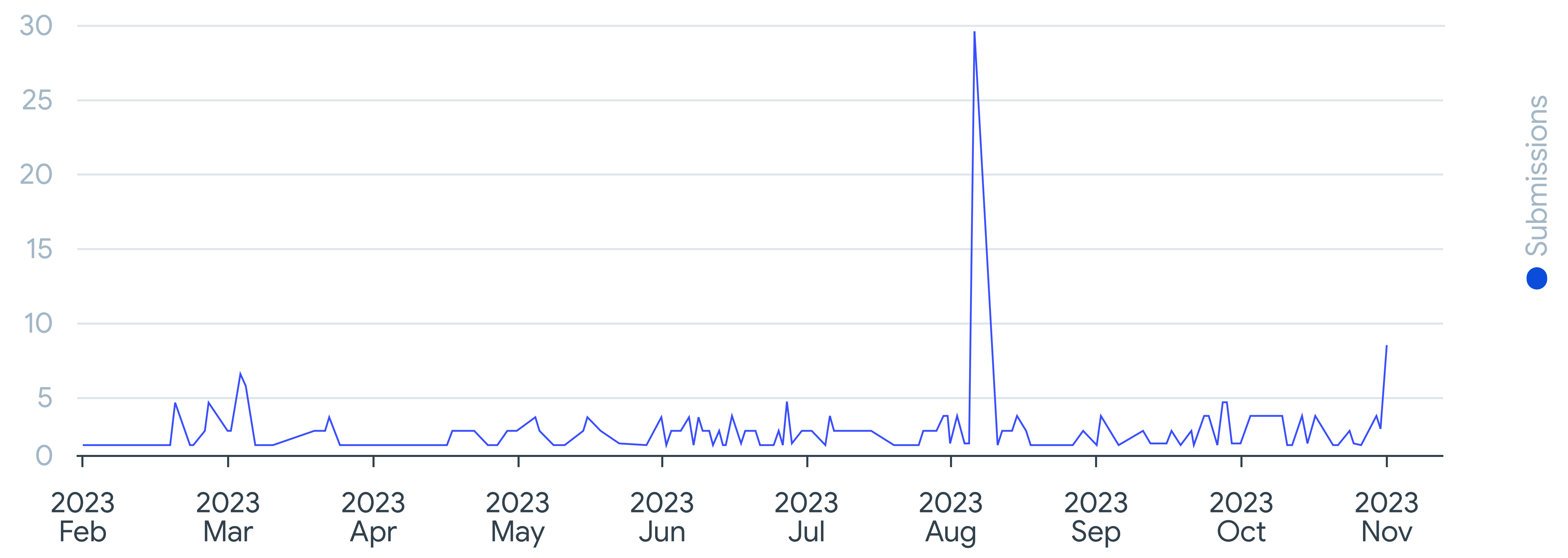


Fig 7

Timeline of malicious samples interacting with openAI API

Most of these files are Windows executables, however, Microsoft Office is the second most frequent file type, followed by Android. Some of the suspicious Microsoft Office documents seem to interact with OpenAI's API by using some extensions (such as VBA-WEB or SEMTools), which seems to correspond to August's peak in the previous chart. Besides this, we found traces of the DarkComet RAT samples embedding OpenAI's API URL in its code. The true nature of this interaction is still unclear.

In addition, we found some examples of malware we suspect could have been AI-generated, although we believe these are experiments conducted most likely by security researchers. In some of them, the prompt used to create the script is included in the sample.

```
@SkeletonMan - Write a Powershell script that disables user input, changes the
background to Nyancat, disables all network/internet access, then encrypts all
files in the Desktop, Documents, Downloads, Pictures, and Videos directories, then
changes the user's password to a random string that is not given, then disables
all fans in the system, then overclocks the CPU by 200%, overclocks the GPU by
200%, overvolts the RAM by 150%, then stresses the hardware to maximum
```

Fig 8

Prompt to produce a malicious script

The previous prompt produces an impossible PowerShell script that uses both Windows and Linux resources to accomplish the instructed outcome.

Final thoughts

The implementation of different AI engines in VirusTotal gave us an unique opportunity to better understand their capabilities in a real environment. There is no doubt that everything will continue to evolve quickly - but as of today, AI engines showed incredible potential for many analysis tasks, including some of the most time consuming tasks such as deobfuscation and describing suspicious behavior.

AI engines being able to offer a comprehensive explanation of code is probably one of the most relevant changes for malware analysts, as it can be challenging to understand why other security tools provide a particular verdict. Being able to go through the reasoning behind a verdict supposes a great advantage for any analyst, helping understanding why this decision was made and making it possible to challenge it.

AI's ability to analyze, identify and explain the intent of malicious scripts is also beneficial to the future of malware analysis - promising faster, more accurate results, and opening up the field of malware analysis to those without highly specialized knowledge or experience. Given Europe is experiencing a shortage of up to 500,000 cybersecurity professionals, AI's ability to make malware analysis faster, more accurate and more accessible could have a real impact - if harnessed correctly - on tackling some of Europe's cybersecurity threats.

We observed interesting results when it comes to CVE and obfuscation detection, notably with regard to AI's stronger performance over other techniques. We also believe that the "universal" code analysis capabilities AI engines demonstrated help avoiding blindspots other security solutions might have at the moment, especially for all non-endpoint detection and protection. File type detection for text files is another unexpected advantage.

When it comes to improving AI engines used for malware analysis, it seems clear they will benefit from a richer context. A more customized prompting could also help align analysis with particular criteria where we now see AI engines and traditional solutions might disagree.

In terms of using AI engines for malware generation, or during malware execution, we couldn't find any solid evidence. There is no doubt of the interest attackers might have in these technologies, including an underground market for "uncensored" AI engines. We expect to see improvements in social engineering deliverable quality, although developing good techniques to prove this is based on the use of AI engines will take some time.

Summarizing, we can see clear improvements in terms of malware analysis for:

- Different angle on malware detection, from a binary verdict to a detailed explanation.
- Enormous timesaving for script analysis, including deobfuscation.
- Powerful tool for detection and analysis of malicious tool sets overlooked by traditional security products.

We will keep studying how AI can help us stop malware, and keep an eye open this technology is not used in any malicious way.

Join the discussion @virustotal

Find out more at virustotal.com

Google