TryHackMe Advent of Cyber 2025 Day 18 Challenge Report

Obfuscation Techniques & PowerShell Analysis

1. Executive Summary

This report documents the completion of Day 18 of the TryHackMe Advent of Cyber 2025 event. The challenge focused on understanding obfuscation techniques used by threat actors to evade detection and delay forensic analysis. Successfully analyzed the SantaStealer.ps1 PowerShell malware, deobfuscated Base64-encoded Command and Control (C2) URLs, and obfuscated API keys using XOR operations. Retrieved both challenge flags demonstrating practical understanding of obfuscation detection, CyberChef operations, and PowerShell script manipulation.

2. Understanding Obfuscation

2.1 What is Obfuscation?

Obfuscation is the practice of making data deliberately hard to read and analyze. Threat actors employ obfuscation techniques to evade basic security detection mechanisms and significantly delay forensic investigations. Unlike encryption (designed for confidentiality with reversibility via keys), obfuscation focuses on making code or data confusing while maintaining functionality.

2.2 Why Attackers Use Obfuscation

- **Evade Detection:** Bypass signature-based security tools
- **Delay Analysis:** Increase time required for incident response
- **Hide Indicators:** Conceal malicious strings, URLs, IPs
- **Bypass Filters:** Circumvent keyword-based blocking

3. Common Obfuscation Techniques

3.1 ROT1 Cipher

ROT1 is a simple Caesar cipher that shifts each letter forward by 1 position in the alphabet:

- a → b, b → c, c → d, etc.
- z → a (wraps around)

Example:

```
Original: carrot coins go brr ROT1: dbsspu dpjot hp css
```

Detection: Common words look 'one letter off', spaces remain unchanged

3.2 ROT13 Cipher

ROT13 shifts letters by 13 positions - exactly halfway through the 26-letter alphabet:

- a → n, b → o, c → p, etc.
- Property: Applying ROT13 twice returns original text

Example:

```
Original: the quick brown fox ROT13: gur dhvpx oebja sbk
```

Detection: Look for three-letter words - 'the' becomes 'gur', 'and' becomes 'naq'

3.3 Base64 Encoding

Base64 converts binary data into ASCII text using 64 printable characters (A-Z, a-z, 0-9, +, /):

- Commonly used for data transmission
- NOT encryption - easily reversible
- Often used in malware to hide strings

Detection Characteristics:

- Long strings of alphanumeric characters
- May contain + or / characters
- Often ends with = or == padding

3.4 XOR Operation

XOR (Exclusive OR) is a mathematical operation used for encryption/obfuscation:

- Each byte combined with key using XOR operation
- Reversible: XOR(XOR(data, key), key) = data
- Results often contain uncommon symbols

Example with CyberChef:

Input: carrot supremacy

XOR Key: 0x61 (hex) = 'a'

```
Output: ikxxe~*yzxogkis!
```

Detection: Random-looking symbols, same length as original, may show repeating patterns if short key reused

3.5 Layered Obfuscation

Threat actors often combine multiple techniques to increase analysis difficulty:

Example Chain:

- Original script → gzip compression
- Compressed data → XOR with key 'x'
- XOR'd data → Base64 encoding

Reversal Process: Apply operations in REVERSE order: Base64 decode → XOR with 'x' → gzip decompress

4. Detection Patterns

Visual clues help identify obfuscation techniques:

4.1 Quick Identification Guide

- **ROT1:** Words one letter off, spaces unchanged, easy to spot
- **ROT13:** Three-letter words like 'the'→'gur', 'and'→'naq'
- **Base64:** Long alphanumeric strings, often ending =, ==
- **XOR:** Random symbols, same length as original, potential repeating patterns

4.2 CyberChef Magic Operation

When obfuscation technique is unknown, CyberChef's Magic operation automatically tests common decoders:

- Displays multiple potential results
- Enable 'Intensive mode' for thorough testing
- Limitation: Won't catch custom XOR keys or unusual layers

5. Challenge Investigation

5.1 Scenario Overview

McSkidy detected suspicious phishing email containing PDF with embedded PowerShell script. Script extracted to isolated VM for analysis. Investigation required deobfuscation to understand malicious functionality.

Script: SantaStealer.ps1 (located on VM Desktop)

5.2 Part 1: C2 URL Deobfuscation

Objective:
Deobfuscate Command and Control URL to reveal attacker infrastructure

Step 1: Open Script

- Located SantaStealer.ps1 on Desktop
- Opened in Visual Studio (double-click, wait for load)
- Navigated to 'Start here' section

Step 2: Identify Encoded Data

- Located $C2B64 variable containing Base64 string
- Copied Base64-encoded value

Step 3: Decode with CyberChef

- Opened CyberChef (online or AttackBox bookmark)
- Pasted Base64 string into Input field
- Dragged 'From Base64' operation to Recipe
- Clicked 'BAKE!' to decode

Step 4: Update Script

- Copied decoded URL from CyberChef output
- Pasted into $C2 variable placeholder in script
- Saved file (Ctrl+S)

Step 5: Execute Script

- Opened PowerShell (Start → type 'PowerShell' → Enter)
- Navigated to Desktop: cd .\Desktop\
- Executed script: .\SantaStealer.ps1
- Script validated C2 URL and displayed flag

Flag Retrieved: THM{C2_De0bfuscation_29838}

5.3 Part 2: API Key Obfuscation

Objective:

Obfuscate attacker's API key using XOR to evade detection

Step 1: Locate API Key

1. Found 'Part 2' section in SantaStealer.ps1
2. Identified $ApiKey variable with plaintext value
3. Copied API key string

Step 2: Apply XOR in CyberChef

4. Pasted API key into CyberChef Input
5. Dragged 'XOR' operation to Recipe
6. Set Key to: 0x37
7. Ensured dropdown next to Key set to: HEX
8. Clicked 'BAKE!' to apply XOR

Step 3: Convert from Hex

9. Added 'From Hex' operation to Recipe
10. Obtained obfuscated API key in Output

Step 4: Update and Execute

11. Copied obfuscated value from CyberChef
12. Pasted into appropriate variable in script
13. Saved file
14. Re-executed script in PowerShell
15. Script validated obfuscation and displayed second flag

Flag Retrieved: THM{API_Obfusc4tion_ftw_0283}

6. Key Skills Developed

- Obfuscation technique identification
- Pattern recognition for encoded data
- Base64 encoding/decoding
- XOR operation application
- CyberChef recipe construction
- PowerShell script analysis
- Malware deobfuscation techniques
- Visual Studio Code usage

7. Conclusion

Day 18 of the TryHackMe Advent of Cyber 2025 provided comprehensive training in obfuscation techniques used by threat actors. Successfully analyzed PowerShell malware (SantaStealer.ps1), identified Base64-encoded Command and Control infrastructure, and applied XOR obfuscation to API keys.

The challenge demonstrated practical application of CyberChef for both deobfuscation (revealing attacker infrastructure) and obfuscation (mimicking attacker techniques). Understanding common obfuscation patterns - ROT ciphers, Base64, XOR, and layered approaches - is essential for malware analysis, incident response, and threat hunting. These skills enable security analysts to quickly identify and reverse attacker attempts to evade detection, significantly reducing investigation time and improving defensive capabilities.

Challenge Status: **COMPLETED ✓**