PUCE Virtual

DOCUMENTO DE PROFUNDIZACIÓN

Programación III

Estudio de caso 2: Análisis de Movimiento Lateral mediante Grafos No Ponderados

Estudio de caso 2: Análisis de Movimiento Lateral mediante Grafos No Ponderados

Contexto

La multinacional "TechGlobal Corp" posee una infraestructura de red híbrida con miles de terminales (endpoints), servidores y dispositivos móviles. En este entorno, las relaciones de comunicación no son lineales, sino que forman una malla compleja de interacciones diarias.

Problema

Tras una auditoría, se detectó que una vez que un atacante logra comprometer una terminal de usuario (punto de entrada), puede realizar movimiento lateral para alcanzar servidores críticos de bases de datos. El problema es que las listas de control de acceso (ACL) tradicionales no permiten visualizar el "camino de menor resistencia" que un atacante podría seguir aprovechando las confianzas existentes entre máquinas.

Solución Propuesta

Modelar la red completa como un **Grafo No Ponderado**.

- **V (Vértices):** Representan cada dispositivo o usuario en la red.
- **E (Aristas):** Representan una conexión o permiso de comunicación activo entre dos nodos. Al ser un grafo no ponderado, nos enfocamos en la **conectividad y la topología**, tratando todas las conexiones como igualmente posibles para un atacante.

Implementar un sistema de análisis de **Centralidad y Alcance** basado en grafos. El objetivo es identificar "nodos articuladores" o "puentes" que, si son comprometidos, darían al atacante acceso a gran parte de la red de manera inmediata.

Estrategia:

La estrategia utiliza algoritmos de recorrido de grafos:

1. Mapeo de Relaciones: Extraer logs de Active Directory y flujos de red para construir el grafo.
2. Cálculo de Centralidad de Intermediación (Betweenness Centrality): Identificar qué nodos actúan como puentes críticos entre diferentes segmentos de red.
3. Análisis de Componentes Conectados: Determinar si la red está demasiado "abierta", permitiendo que cualquier nodo llegue a cualquier otro en pocos saltos.

Implementación

Se utiliza la librería NetworkX en Python para procesar el grafo de la infraestructura:

- **Algoritmo BFS (Breadth-First Search):** Para calcular la distancia mínima (en saltos) desde cualquier nodo infectado hasta el servidor central.
- **Detección de Comunidades:** Para segmentar la red lógicamente y verificar que los grupos (ej. Finanzas vs. Invitados) no tengan aristas (conexiones) no autorizadas entre ellos.

Implementación en Python

```python
import networkx as nx
import matplotlib.pyplot as plt

# 1. Creación del Grafo No Ponderado
G = nx.Graph()

# Definimos los nodos (Dispositivos/Usuarios)
nodos = [
    "PC-Recepcion", "PC-Ventas", "PC-RRHH",
    "Switch-Core", "Server-BasesDeDatos", "Server-Web",
    "Admin-IT", "PC-Contabilidad"
]
G.add_nodes_from(nodos)

# 2. Definición de Aristas (Conexiones/Confianzas)
# Al ser no ponderado, solo indicamos que la conexión EXISTE
conexiones = [
    ("PC-Recepcion", "Switch-Core"),
    ("PC-Ventas", "Switch-Core"),
    ("PC-RRHH", "Switch-Core"),
    ("Switch-Core", "Server-Web"),
    ("Switch-Core", "Admin-IT"),
    ("Admin-IT", "Server-BasesDeDatos"), # Camino legítimo
    ("PC-Contabilidad", "Server-BasesDeDatos"), # ¡Vulnerabilidad detectada!
    ("PC-Contabilidad", "Switch-Core")
]
G.add_edges_from(conexiones)
```

```python
# 3. Análisis de Movimiento Lateral
# Supongamos que "PC-Recepcion" es comprometida por un Ransomware
inicio = "PC-Recepcion"
objetivo = "Server-BasesDeDatos"

try:
    # Algoritmo BFS para encontrar la ruta con menos saltos
    ruta_ataque = nx.shortest_path(G, source=inicio, target=objetivo)
    print(f"⚠️ Alerta: Ruta de movimiento lateral detectada ({len(ruta_ataque)-1} salto
    print(" -> ".join(ruta_ataque))
except nx.NetworkXNoPath:
    print("✅ El servidor crítico está aislado de este segmento.")
```

```python
# 4. Cálculo de Importancia (Centralidad)
# Identifica qué nodos son críticos para la conectividad de la red
centralidad = nx.betweenness_centrality(G)
nodo_critico = max(centralidad, key=centralidad.get)
print(f"\n🔍 Nodo estructuralmente más crítico: {nodo_critico}")

# 5. Visualización
plt.figure(figsize=(10, 6))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=2000, font_size=10)

# Resaltar la ruta del ataque en rojo
path_edges = list(zip(ruta_ataque, ruta_ataque[1:]))
nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color='r', width=3)

plt.title("Visualización de Ruta de Movimiento Lateral en la Red")
plt.show()
```

Resultados

Identificación de Riesgos: Se descubrió que el 15% de las estaciones de trabajo tenían acceso directo a la red de servidores debido a configuraciones de legado.

Optimización de Microsegmentación: Se eliminaron aristas innecesarias en el grafo, reduciendo el "diámetro" del grafo de la red y limitando el radio de explosión de un ataque.

Simulación de Ataques: El equipo azul (Blue Team) pudo predecir las rutas más probables de un atacante antes de que ocurriera un incidente.

Impacto en Ciberseguridad

El enfoque de grafos permite una visión **holística y relacional**:

- **Visualización de la Superficie de Ataque:** Los administradores pueden ver literalmente "los caminos" que el malware seguiría (gusanos de red).
- **Endurecimiento de Red (Hardening):** Priorización de parches en los nodos con mayor grado de centralidad, ya que su compromiso es más peligroso para la organización.

Reflexión Técnica

En un grafo no ponderado, la métrica clave es la distancia geodésica (el número mínimo de saltos). A diferencia de los modelos ponderados que miden latencia o ancho de

banda, aquí lo que importa es la topología de la confianza. Si existe una arista entre un usuario y un servidor, la seguridad depende enteramente de la autenticación; si esa arista no debería existir, el grafo revela una vulnerabilidad de diseño estructural que los firewalls perimetrales suelen pasar por alto.