Root Me
Título: RootMe
Descrição: A CTF é para iniciantes — consegue me rootar?
Objetivo: Reconhecimento, upload malicioso e elevação de privilégios — obter user.txt e root.txt.
Tipo: Boot to Root / Web
Nível: Fácil

Reconhecimento

Começo esta CTF rodando o Nmap para mapear portas e serviços com nmap -sV <ip>. Esse passo inicial serve para entender quais serviços estão expostos ao público e suas versões — informação útil porque versões antigas costumam ter vulnerabilidades conhecidas. Enquanto verifico portas abertas, também abro a aplicação web no navegador e inspeciono os headers HTTP para identificar servidor, linguagem ou pistas sobre frameworks; cabeçalhos e versões às vezes entregam atalhos úteis. Não encontrando nada óbvio na interface, sigo aumentando a superfície de ataque pela enumeração de diretórios.

O scan do Nmap retorna duas portas abertas: 80 (o site) e 22 (SSH em um servidor Linux — Ubuntu). O Apache apresenta versão 2.4.41. Como não temos credenciais nem acesso via SSH, concentramos esforço no serviço web. Para descobrir caminhos escondidos e arquivos administrativos, uso ferramentas de brute-force de diretórios; por exemplo, gobuster dir -u http://<ip> -w <wordlist> percorre listas de nomes comuns e revela pontos que o site não linka diretamente. Com isso encontramos um diretório /panel/ e uma pasta uploads/. Ao acessar uploads/ aparece uma página de upload de arquivos para o servidor Apache — ouro puro. Um endpoint de upload é perigoso porque, se não houver validação correta do tipo e do conteúdo dos arquivos, é possível colocar um arquivo que será interpretado pelo servidor, criando uma entrada direta no sistema.

Conseguindo uma entrada

Com o ponto de upload identificado, tento enviar um reverse shell em PHP — um script que, quando executado pelo servidor, se conecta de volta para a minha máquina e me dá um terminal remoto. Normalmente, servidores aplicam filtros que verificam extensão de arquivo ou conteúdo para bloquear uploads executáveis. O site inicialmente bloqueia a tentativa, então começo a testar variações: mudo o nome do arquivo, insiro comentários, altero a extensão e observo as respostas do servidor. Essas pequenas mudanças visam burlar validações fracas que confiam apenas na extensão ou fazem checagens superficiais.

Eventualmente descubro que arquivos com extensão .php5 são aceitos pelo mecanismo de upload. Essa aceitação pode ocorrer porque o servidor ou a validação só verifica “.php” explícito ou porque a configuração do Apache reconhece .php5 como interpretável. Aproveitando isso, envio o reverse shell configurado para se conectar ao meu listener. No meu lado, preparo um listener

com nc -lvnp 4444 para escutar a conexão de retorno; “nc” (netcat) é uma ferramenta simples e confiável para receber reverses. Faço o upload do reverse shell.php5 e, com o Netcat escutando, recebo a conexão — a aplicação web processou o arquivo e a máquina alvo estabeleceu a conexão de volta, abrindo um terminal remoto do Apache. Essa etapa demonstra como um upload mal filtrado pode quebrar a fronteira entre web e sistema operacional.

Dentro do Apache

A conexão inicial via Netcat funciona, mas é limitada: não tem recursos típicos de um terminal interativo (como histórico com as setas, controle de trabalho, tty completo), o que atrapalha exploração avançada. Para trabalhar melhor, tento estabilizar a shell, transformando-a em um pseudo-terminal mais funcional caso o Python 3 esteja presente na máquina, pois o Python pode invocar um shell TTY que permite usar editores, rodar comandos interativos e melhorar a navegação pelo sistema. Com esse terminal mais amigável, começo a procurar a primeira flag: busco por arquivos chamados user.txt em todo o sistema, ignorando mensagens de erro para não perder tempo com diretórios inacessíveis. Encontro a flag em /var/www/user.txt — era esperado que a flag do usuário estivesse em diretórios relacionados ao serviço web, já que conseguimos entrar por ali. Coletar essa flag confirma o acesso e permite focar na próxima etapa: escalar privilégios.

Escalando privilégios

Com acesso como o usuário do serviço web (não como root), procuro vetores clássicos de elevação de privilégios. Um dos pontos que sempre verifico são executáveis com o bit SUID ativo, porque eles rodam com as permissões do dono (normalmente root) e, se forem programas que permitem execução de código ou interpretação, podem ser abusados. Uso uma busca por arquivos com permissão SUID para listar candidatos. Entre os resultados, /usr/bin/python2.7 se destaca: ter um interpretador com SUID é arriscado, pois o Python permite executar comandos arbitrários e pode ser usado para spawnar um shell que mantenha privilégios elevados.

Sabendo disso, executo o interpretador Python com uma instrução curta que substitui o processo atual por um shell privilegiado e preserva os privilégios herdados. Com essa técnica, o processo que eu controlo passa a ter permissões de root. Ao obter esse shell com permissões máximas, faço uma busca pela flag final usando os nomes padrões; encontro root.txt em /root/root.txt. A presença de binários SUID mal configurados costuma ser consequência de manutenção negligente ou de configurações herdadas, e históricas como essa explicam por que auditorias de permissões são críticas em sistemas reais.

Conclusão

Em resumo, o ataque foi uma progressão lógica e incremental: primeiro entender a superfície de ataque, depois encontrar um ponto de entrada viável, explorar esse ponto para executar código e finalmente abusar de má

configuração de permissões para escalar até root. O vetor inicial foi uma funcionalidade de upload web que aceitou um arquivo com extensão .php5; a exploração envolveu envio de um reverse shell PHP e recepção da conexão no meu listener. Após estabilizar o terminal, coletei user.txt em /var/www/user.txt. Para elevação de privilégios, identifiquei binários SUID e abusei de /usr/bin/python2.7 para spawnar um shell com privilégios preservados, obtendo assim root e a flag em /root/root.txt.

FIM.